



NVIDIA.

OpenACC

# Welcome to the Megatron GPT bootcamp

Application Expert -- Quantum Chemistry

Dr. Johan Raber

Director of the EuroCC National Competence Center Sweden (ENCCS)

Dr. Lilit Axner



# Day 1 - intro & Getting Started

## Outline of Day 1 ( +3 hour )

Day 1: October 25, 2021 (9:00 AM to 12:00 PM CEST)

**09:00 AM - 09:15 AM: Welcome and opening note ( Lilit )**

09:15 AM - 09:30 AM: Bootcamp Overview

09:30 AM - 10:00 AM: intro to NV SuperPOD (Dai)

10:00 AM - 10:20 AM: [demo] Multi-nodes Megatron training

10:20 AM - 10:40 AM: Team-Up time

10:40 AM - 11:40 AM: Challenge overview & hands-on (Lab)

11:40 AM - 12:00 AM: Connect with the expert - coffee corner

[ for those who finished the challenge early ]

11:40 AM - 12:00 PM: Discussion & What's up next day

---

# INTRODUCING THE TA

Introducing the lead TAs



J-C Vasnier

(Day 1)



Xianchao Wu

(Day 1,2,3)



Meriem Bendris

(Day 2,3)

# INTRODUCING THE TA

Introducing the TAs - ATOS



Giorkallos, Alexis

(Day 1,2,3)



Tournadre, Maxime

(Day 1,2,3)



Ulker, Erkan

(Day 1,2,3)

# INTRODUCING THE SPEAKERS OF THE DAY

Day 1's speakers - Dai Yang and Bruno Charrier



Dai Yang

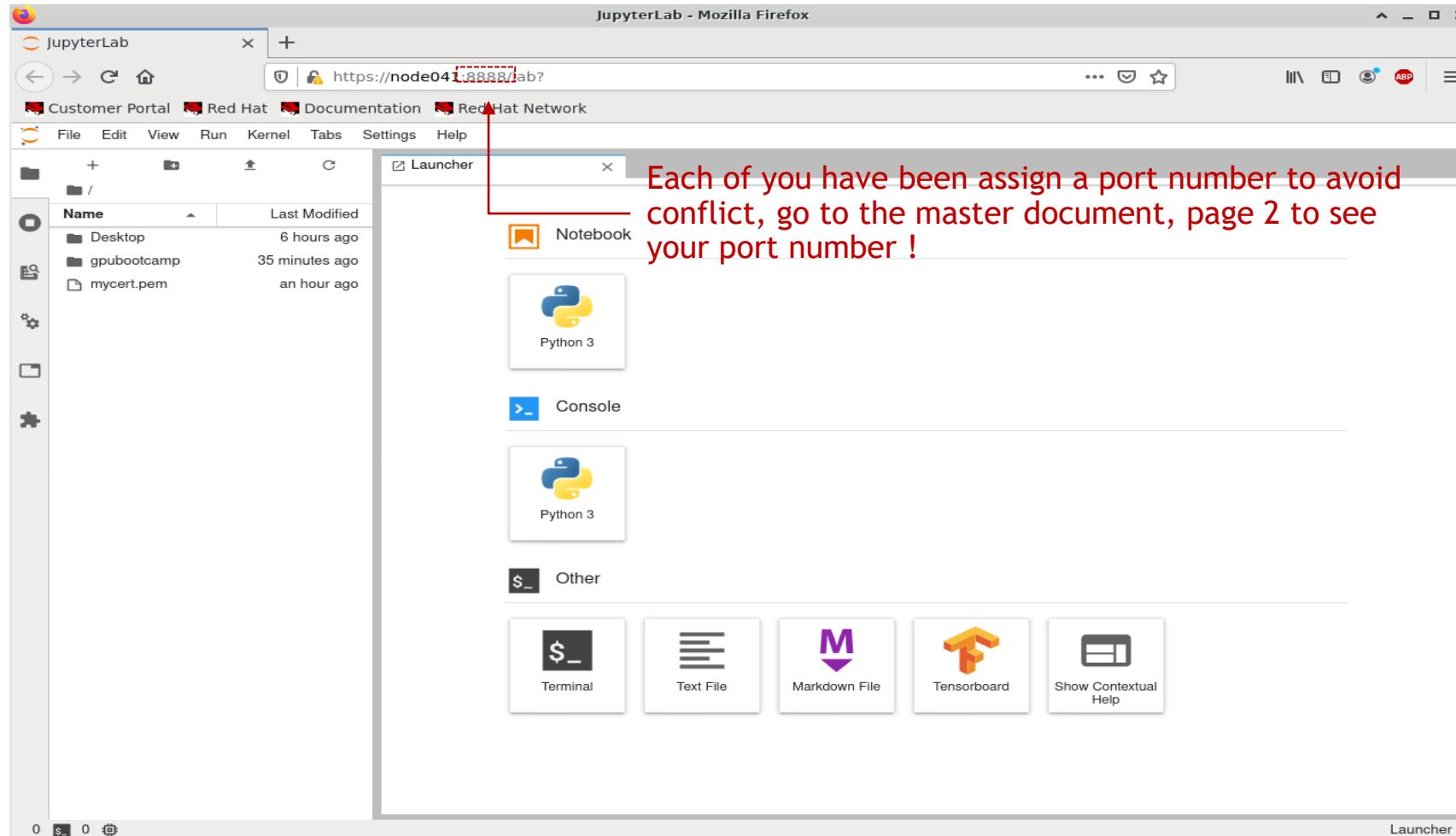


Bruno Charrier

# PREPARE FOR DAY 2'S ENVIRONMENT

Download the instruction from [shared google drive](#) :

Following the step-by-step instruction from that PDF file to access the jupyter lab



# LOGISTICS OF THE BOOTCAMP

Zoom functions | Slack channels | check cluster access

Join Slack : <https://tinyurl.com/4and8wsy>

[Master doc](#)

## Login (or access) nodes

Typically a few nodes accessible from the internet. This where you end up when accessing the cluster via SSH or [ThinLinc](#) for instance, and is where you transfer files to. These nodes are not meant for heavy computations as they are shared by all users accessing the cluster, but are used to request compute resources (and build code, edit scripts etc) for compute intensive jobs. On berzelius these nodes are called [berzelius1.nsc.liu.se](#) and [berzelius2.nsc.liu.se](#).

<https://www.nsc.liu.se/support/systems/berzelius-getting-started/>



## General links:

- Link to the Master document: <https://tinyurl.com/v85dz3sz>
- GPU Hackathons Website: <https://gpuhackathons.org/>
- Agenda: <https://gpuhackathons.org/event/enccs-nsc-megatron-bootcamp>
- Slack Channel: <https://tinyurl.com/4and8wsy>
- Zoom information: <https://zoom.us>

Meeting ID: 856 7289 9555  
Passcode: 021869

- Cluster access: <https://tinyurl.com/sticypxy> (9:00-15:00)
- Port allocations: Check the 2nd page
- Survey: <https://forms.gle/G2WUWcPuiQ6DUTH79>

## General Guidance

- 🌟 Be kind. Be understanding. Be flexible.
- 📄 Slides will be available after the event.
- 😊 Keep yourself muted when not speaking to minimise background noise.
- 📺 Zoom chat is disabled; general chat in the **ENCCS & NSC Megatron Bootcamp 2021** Slack workspace.
- 💚 Introduce yourself to fellow participants, tell us about your expertise and interests in the

Applications

Sun 26 Sep, 13:43 Zenodia Charpy



Trash



File System



Home



# FROM QUESTIONS TO ANSWERS

In this bootcamp, we are trying to address to these questions

Why should I care about training very large NLP models

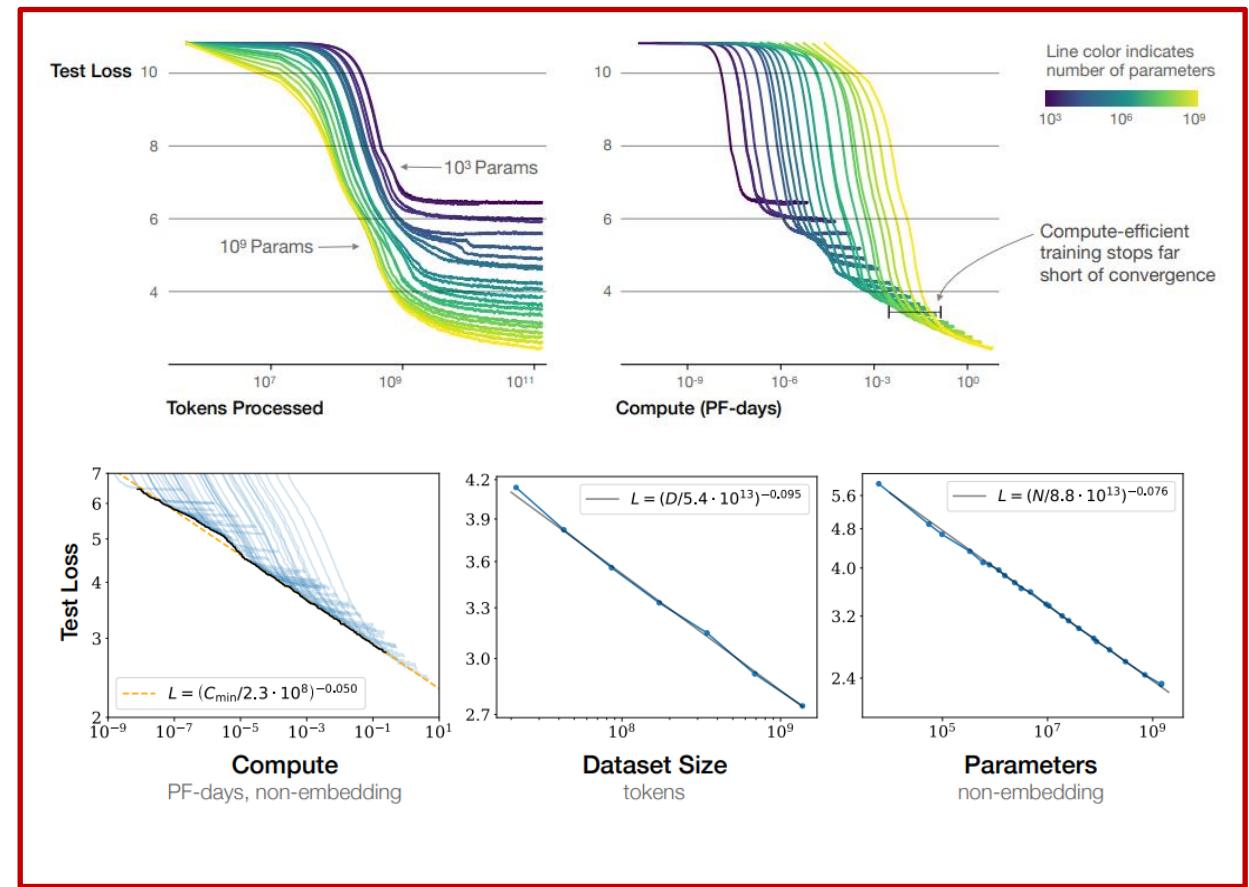
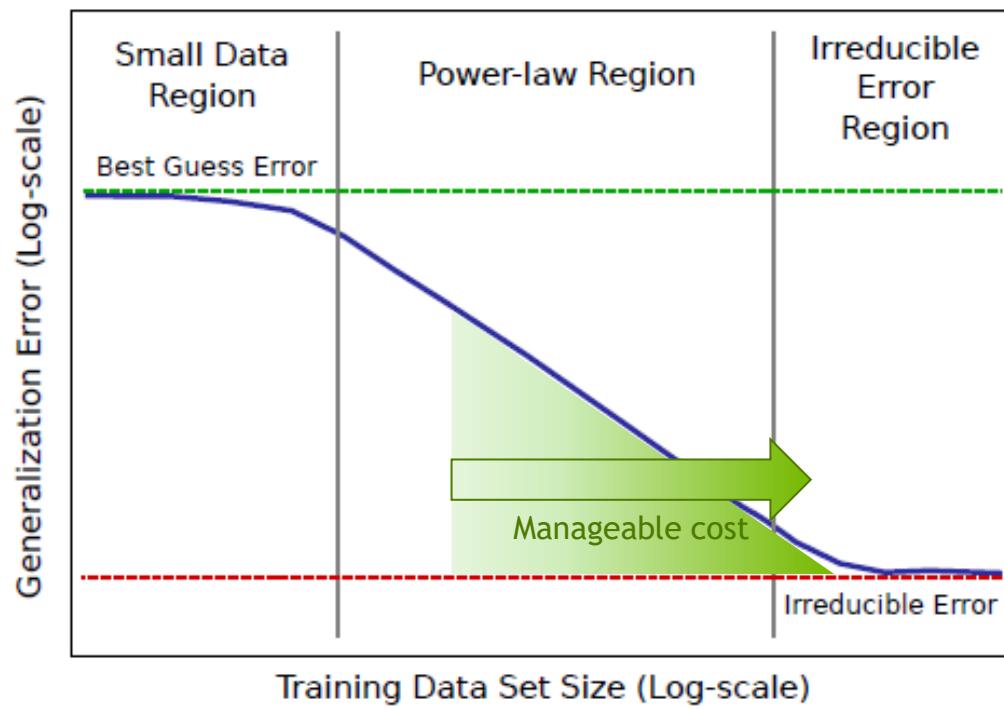
How to kick-start to train large NLP models with my own language /  
own dataset

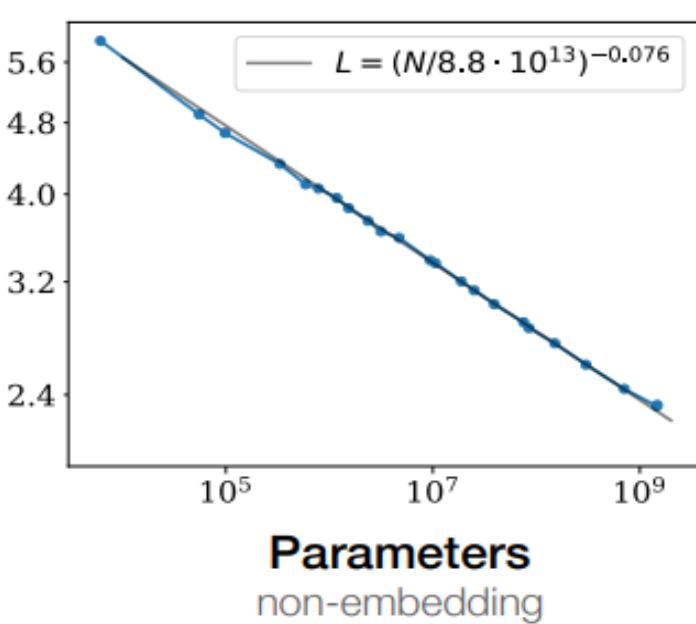
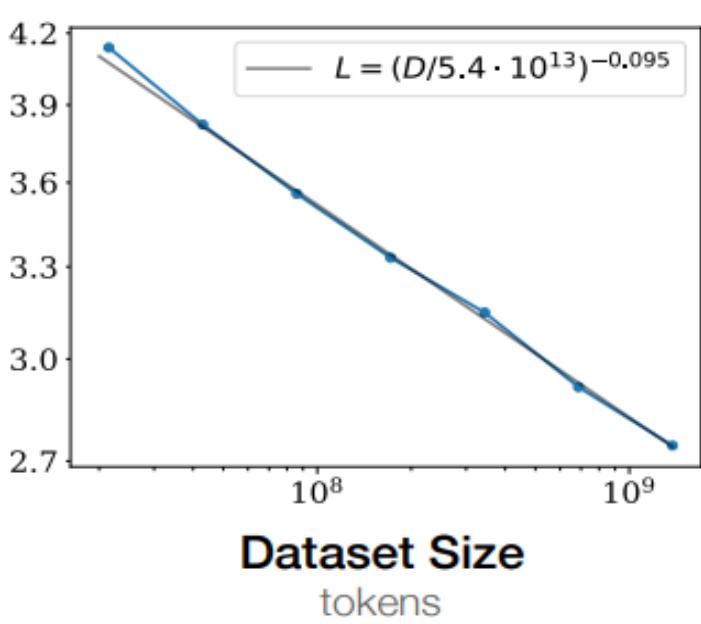
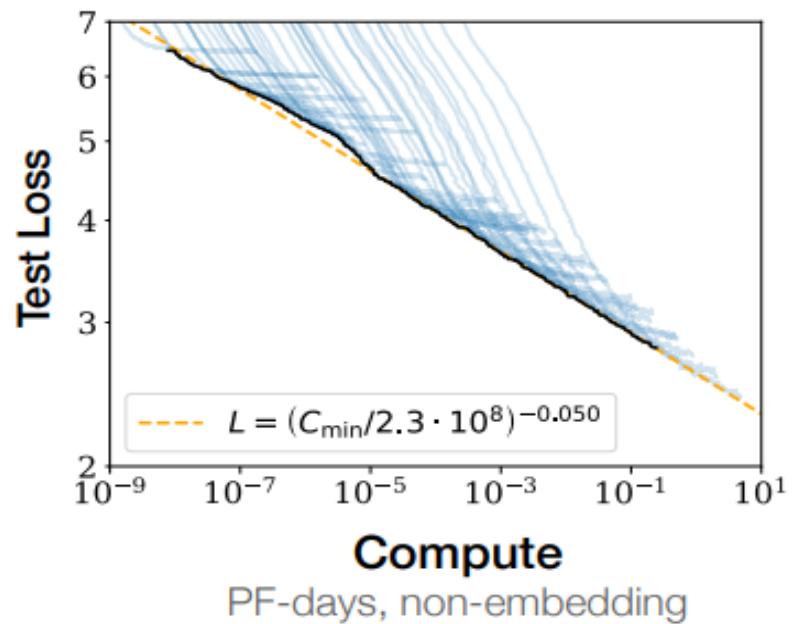
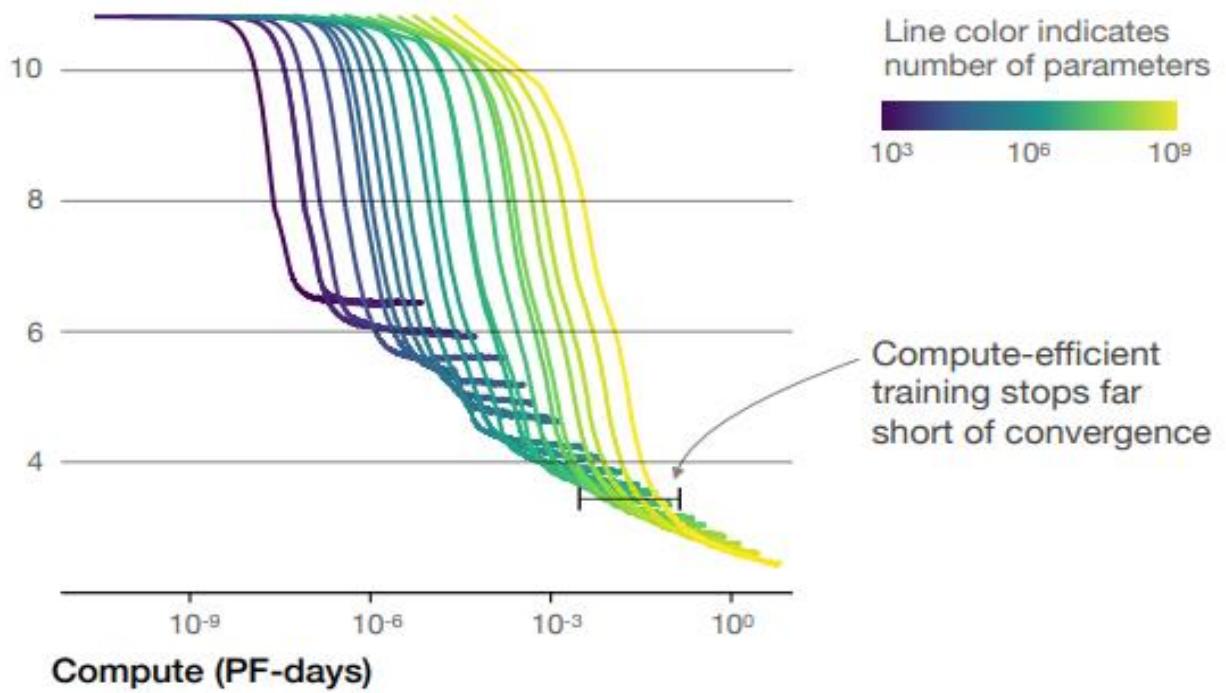
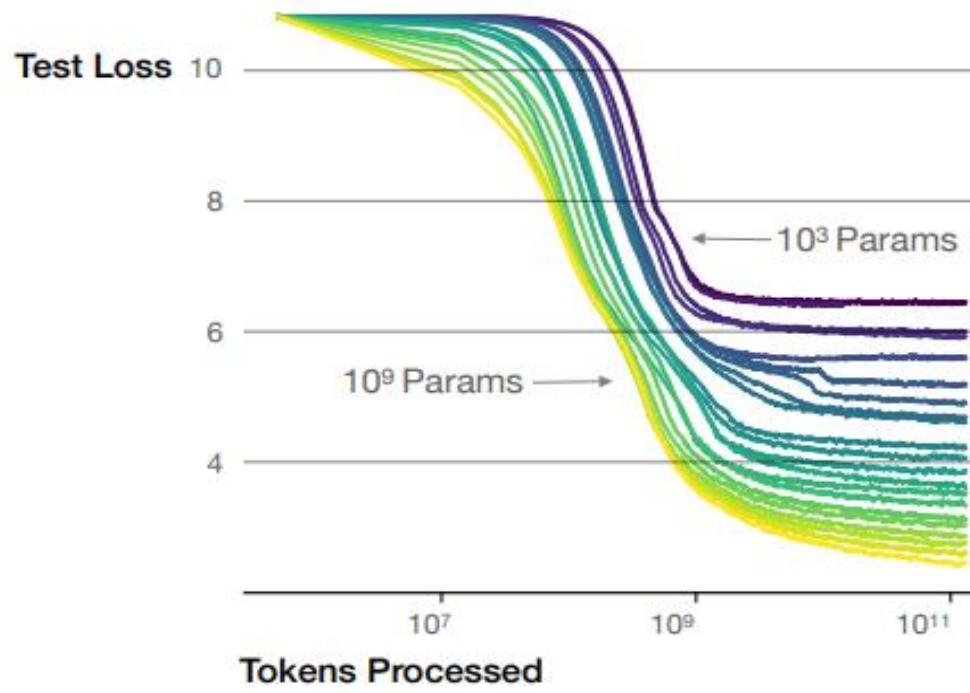


WHY VERY LARGE NLP MODELS ?

# THE PROMISE

Big data + Big model + Big compute





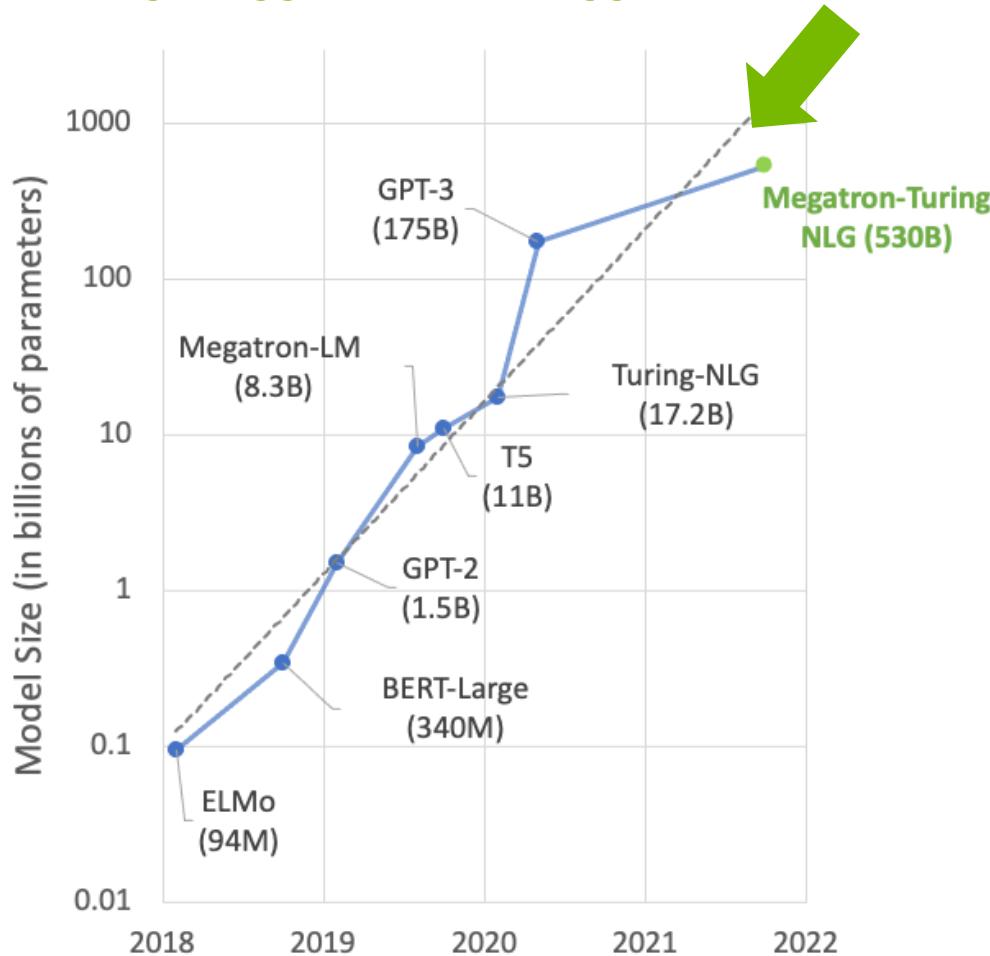
# THE SCALING LAWS

## MOST IMPORTANT!!

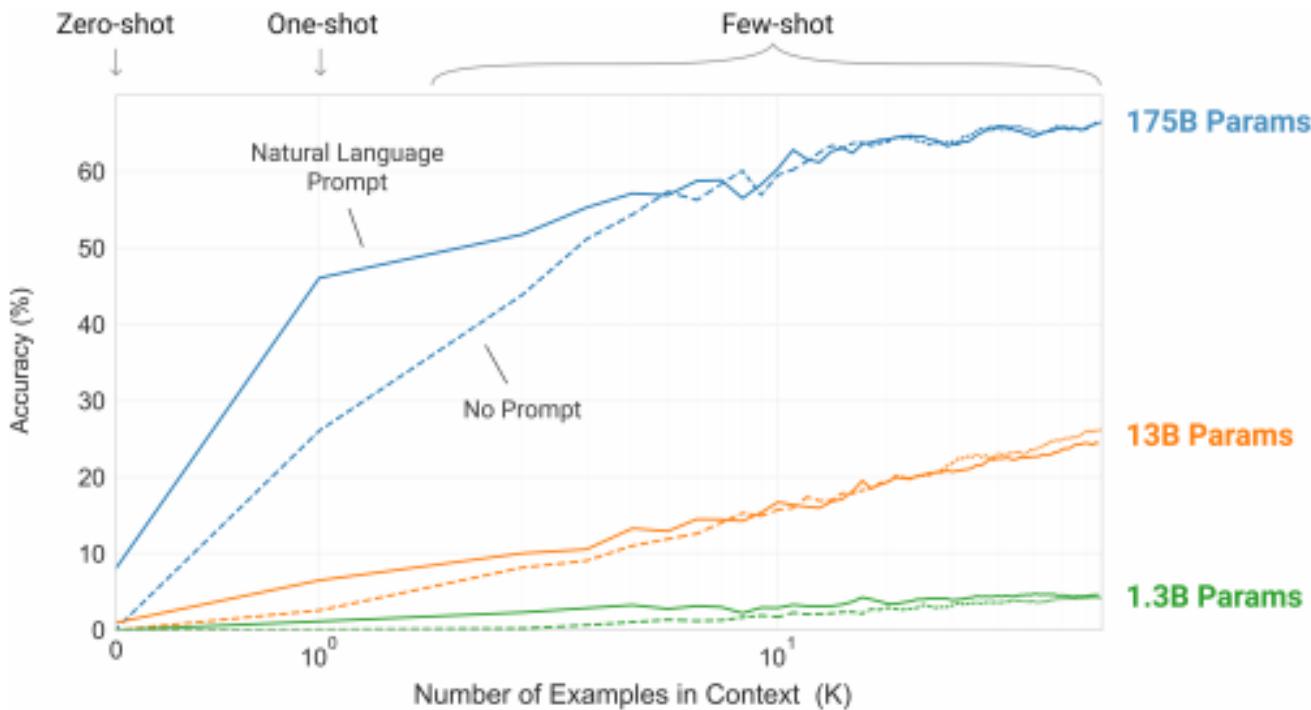
*“... more importantly, we find that the precise architectural hyperparameters are unimportant compared to the overall scale of the language model.”*

# THE SCALING LAWS

Big, Bigger , even Bigger models



# EVEN MORE IMPORTANTLY



**Figure 1.2: Larger models make increasingly efficient use of in-context information.** We show in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description (see Sec. 3.9.2). The steeper “in-context learning curves” for large models demonstrate improved ability to learn a task from contextual information. We see qualitatively similar behavior across a wide range of tasks.

# INGREDIENT : BIG DATA + BIG MODEL + BIG COMPUTE

A photograph of a dark night sky filled with stars. In the foreground, the silhouettes of trees are visible against the dark background. Three bright, glowing blue stars are positioned in the upper half of the image, each connected by a thin white line to a text label:

- The star on the left is labeled "Big Data Volume".
- The star in the middle is labeled "Motivated researchers".
- The star on the right is labeled "Compute Power".



# Day 1 - intro & Getting Started

## Outline of Day 1 ( +3 hour )

**Day 1: October 25, 2021 (9:00 AM to 12:00 PM CEST)**

09:00 AM - 09:15 AM: Welcome and opening note ( Lilit )

### **09:15 AM - 09:30 AM: Bootcamp Overview**

09:30 AM - 10:00 AM: intro to NV SuperPOD (Dai)

10:00 AM - 10:20 AM: [demo] Multi-nodes Megatron training

10:20 AM - 10:40 AM: Team-Up time

10:40 AM - 11:40 AM: Challenge overview & hands-on (Lab)

11:40 AM - 12:00 AM: Connect with the expert - coffee corner

[ for those who finished the challenge early ]

11:40 AM - 12:00 PM: Discussion & What's up next day

---

# OVERVIEW 3 HALF DAYS BOOTCAMP

## objective overview

### Day 1

Familiarize with the SuperPOD environment

Get Megatron to run by default

Bonus session : the man behind Cambridge-1

### Day 2

Intro to Megatron workflow

Profiling

Bonus session : importance of profiling - ask the expert!

### Day 3

Customize Megatron

On scaling model size

Bonus session 1: on deploying large NLP models

Bonus session 2: ask the experts on NVIDIA bigNLP vision



# Day 1 - intro & Getting Started

## Outline of Day 1 ( +3 hour )

**Day 1: October 25, 2021 (9:00 AM to 12:00 PM CEST)**

09:00 AM - 09:15 AM: Welcome and opening note ( Lilit )

09:15 AM - 09:30 AM: Bootcamp Overview

09:30 AM - 10:00 AM: intro to NV SuperPOD (Dai)

10:00 AM - 10:20 AM: [demo] Multi-nodes Megatron training

10:20 AM - 10:40 AM: Team-Up time

10:40 AM - 11:40 AM: Challenge overview & hands-on (Lab)

11:40 AM - 12:00 AM: Connect with the expert - coffee corner

[ for those who finished the challenge early ]

11:40 AM - 12:00 PM: Discussion & What's up next day

---



# Day 2 - Run through Megatron GPT

## Outline of Day 2 ( 5.5 hours lunch break)

09:00 AM - 09:15 AM: Environment prep

09:15 AM - 09:45 AM: About compute

09:45 AM - 10:15 AM:

Lab 1 - Acquire data ( WebScraping )

Lab 1 - 2\_Estimate Time Needed for Compute

10:15 AM - 10:45 AM: Lab 1 - 3\_Megatron's core MPU

10:45 AM - 11:00 AM: Break

11:00 AM - 11:40 AM:

Lab 1 - 4\_GPT Vocab+Merge file

Lab 1 - 5\_data\_preprocessing2mmap

11:40 AM - 12:00 PM: Intro to profiling

12:00 PM - 12:10 PM: Ask the Experts (Robert Dietrich)

12:10 PM - 13:10 PM: Lunch break

13:10 PM - 13:40 PM: Lab 1 - 6\_GPT training config vs GPUs performance

13:40 PM - 14:10 PM: Challenge overview

14:10 PM - 14:30 PM: Discussion & What's up next day



# Day 3 - Customize Megatron for Swedish

## Outline of Day ( 6 hour with lunch break )

09:00 AM - 09:15 AM: Recap and Overview of Day 3

09:15 AM - 09:30 AM: About acquiring your own

09:30 AM - 10:20 AM:

    Lab 2 - 1\_Acquiring data

    Lab 2 - 2\_data cleaning & filter

10:20 AM - 10:30 AM: train own GPT tokenizer + preprocessing

10:30 AM - 11:00 AM: Lab 2 - 3\_train your own GPT tokenizer

11:00 AM - 11:30 AM: Lab 2 - 4\_data preprocessing

11:30 AM - 12:00 PM: Lab 2 - 4 Mini challenge

12:00 PM - 12:30 PM: Lunch Break ( 30 mins )

13:00 PM - 14:00 PM: Lab 2 - 5 Challenge - Go BIG or go home

14:00 PM - 14:30 PM: About deploying Big NLP model with NVIDIA Triton | faster transformer solution

14:30 PM - 15:00 PM: Ask the Experts with NVIDIA data scientists on what's next for NLP

15:00 PM - 15:05 PM: Discussion & Final remarks



LET'S START !



# Day 1 - intro & Getting Started

## Outline of Day 1 ( +3 hour )

**Day 1: October 25, 2021 (9:00 AM to 12:00 PM CEST)**

09:00 AM - 09:15 AM: Welcome and opening note ( Lilit )

09:15 AM - 09:30 AM: Bootcamp Overview

**09:30 AM - 10:00 AM: intro to NV SuperPOD (Dai)**

10:00 AM - 10:20 AM: [demo] Multi-nodes Megatron training

10:20 AM - 10:40 AM: Team-Up time

10:40 AM - 11:40 AM: Challenge overview & hands-on (Lab)

11:40 AM - 12:00 AM: Connect with the expert - coffee corner

[ for those who finished the challenge early ]

11:40 AM - 12:00 PM: Discussion & What's up next day

---



# Day 1 - intro & Getting Started

## Outline of Day 1 ( +3 hour )

**Day 1: October 25, 2021 (9:00 AM to 12:00 PM CEST)**

09:00 AM - 09:15 AM: Welcome and opening note ( Lilit )

09:15 AM - 09:30 AM: Bootcamp Overview

09:30 AM - 10:00 AM: intro to NV SuperPOD (Dai)

**10:00 AM - 10:20 AM: [demo] Multi-nodes Megatron training**

10:20 AM - 10:40 AM: Team-Up time

10:40 AM - 11:40 AM: Challenge overview & hands-on (Lab)

11:40 AM - 12:00 AM: Connect with the expert - coffee corner

[ for those who finished the challenge early ]

11:40 AM - 12:00 PM: Discussion & What's up next day

---

A complex network graph is displayed against a dark gray background. The graph consists of numerous small, semi-transparent circular nodes scattered across the frame. These nodes are interconnected by a dense web of thin, light gray lines representing edges. Some nodes are highlighted with a bright lime green color, which are primarily located in the upper left and right quadrants. The overall effect is one of a large, interconnected system or dataset.

ASSETS

Terminal - x\_zench@berzelius001:/proj/guest\_at\_nsc/users/zcharpy

File Edit View Terminal Tabs Help

```
[x_zench@berzelius001 zcharpy]$ pwd
/proj/guest_at_nsc/users/zcharpy
[x_zench@berzelius001 zcharpy]$
```





# MULTI-NODES GPT3 TRAINING DEMO

Terminal - x\_zench@berzelius001:/proj/guest\_at\_nsc/users/zcharpy

```
[x_zench@berzelius001 zcharpy]$ ls
8gpu_run.sh          MultiNodes2N16A100_GPT_train.sh
dataset               mycert.pem
Day1_multinodes_template.sh nsys_Multinodes.sh
demo.sh               output
download_NGCpytorch_sif.sh profiles
download_pytorchsqsh.sh pytorch_21.03.sif
get_certificate.sh    pytorch.sqsh
gpubootcamp           test_nssys.sh
Megatron-LM           test_singularity.sh
[x_zench@berzelius001 zcharpy]$
```

Terminal - x\_zench@berzelius001:~

```
[x_zench@berzelius001 ~]$ watch -n 1 squeue -u x_zench
[x_zench@berzelius001 ~]$
```





# Day 1 - intro & Getting Started

## Outline of Day 1 ( +3 hour )

**Day 1: October 25, 2021 (9:00 AM to 12:00 PM CEST)**

09:00 AM - 09:15 AM: Welcome and opening note ( Lilit )

09:15 AM - 09:30 AM: Bootcamp Overview

09:30 AM - 10:00 AM: intro to NV SuperPOD (Dai)

10:00 AM - 10:20 AM: [demo] Multi-nodes Megatron training

**10:20 AM - 10:40 AM: Team-Up time**

10:40 AM - 11:40 AM: Challenge overview & hands-on (Lab)

11:40 AM - 12:00 AM: Connect with the expert - coffee corner

[ for those who finished the challenge early ]

11:40 AM - 12:00 PM: Discussion & What's up next day

---



# Day 1 - intro & Getting Started

## Outline of Day 1 ( +3 hour )

**Day 1: October 25, 2021 (9:00 AM to 12:00 PM CEST)**

09:00 AM - 09:15 AM: Welcome and opening note ( Lilit )

09:15 AM - 09:30 AM: Bootcamp Overview

09:30 AM - 10:00 AM: intro to NV SuperPOD (Dai)

10:00 AM - 10:20 AM: [demo] Multi-nodes Megatron training

10:20 AM - 10:40 AM: Team-Up time

**10:40 AM - 11:40 AM: Challenge overview & hands-on (Lab)**

**11:40 AM - 12:00 AM: Connect with the expert - coffee corner**

**[ for those who finished the challenge early ]**

11:40 AM - 12:00 PM: Discussion & What's up next day

---



### Task :

you will be given a template script which you should modify in order to run Megatron-LM successfully.

Only modify the parameters in the section denoted ---CHANGE ALLOWED---

# THE CHALLENGE - MAKE IT RUN !

```
#!/bin/bash  
#SBATCH -N 2  
#SBATCH --gpus 16  
#SBATCH --ntasks-per-node 8  
#SBATCH --exclusive  
#SBATCH -A berzelius-2021-43  
#SBATCH --time=3:00:00
```

```
##### Beginning of modifiable section #####
```

```
MY_DIR=  
MYUSR_NAME=  
NUM_LYS=  
HIDDEN_SIZE=  
NUM_ATTN_HEADS=  
SEQ_LEN=  
MAX_POS_EM=  
MICRO_BATCH_SIZE=  
GLOBAL_BATCH_SIZE=  
TENSOR_PARALLEL_SIZE=  
PIPELINE_PARALLAL_SIZE=
```

```
##### end of modifiable sectio, do NOT modify anything below this line #####
```

Modify ONLY this section please !

```
export NNODES=$SLURM_NNODES  
export GPUS_PER_NODE=8  
export MASTER_ADDR=`scontrol show hostnames ${SLURM_NODELIST} | head -1`  
export MASTER_PORT=8890  
export WORLD_SIZE=$((GPUS_PER_NODE*NNODES))
```

```
DIR='/megatron_workspace'  
DATETIME=`date +date_%y-%m-%d_time_%H-%M-%S`  
CHECKPOINT_PATH=$DIR/output/sv_gpt3_ckpt/  
VOCAB_FILE=$DIR/dataset/vocab.json  
MERGE_FILE=$DIR/dataset/merges.txt  
DATA_PATH=$DIR/dataset/SVGPT_32k_text_document
```

```
options="--num-layers ${NUM_LYS} \  
        --hidden-size ${HIDDEN_SIZE} \  
        --num-attention-heads ${NUM_ATTN_HEADS} \  
        --seq-length ${SEQ_LEN} \  
        --max-position-embeddings ${MAX_POS_EM} \  
        --lr 0.00015 \  
        --train-iters 100 \  
        --min-lr 0.00001 \  
        --lr-decay-iters 99 \  
        --lr-warmup-fraction 0.01 \  
        --override-lr-scheduler \  
        --micro-batch-size ${MICRO_BATCH_SIZE} \  
        --global-batch-size ${GLOBAL_BATCH_SIZE} \  
        --tensor-parallel-size ${TENSOR_PARALLEL_SIZE} \  
        --pipeline-parallel-size ${PIPELINE_PARALLAL_SIZE} \  
        --fp16-precision 16" > ./megatron_train.sh
```



# Day 1 - intro & Getting Started

## Outline of Day 1 ( +3 hour )

**Day 1: October 25, 2021 (9:00 AM to 12:00 PM CEST)**

09:00 AM - 09:15 AM: Welcome and opening note ( Lilit )

09:15 AM - 09:30 AM: Bootcamp Overview

09:30 AM - 10:00 AM: intro to NV SuperPOD (Dai)

10:00 AM - 10:20 AM: [demo] Multi-nodes Megatron training

10:20 AM - 10:40 AM: Team-Up time

10:40 AM - 11:40 AM: Challenge overview & hands-on (Lab)

11:40 AM - 12:00 AM: Connect with the expert - coffee corner

[ for those who finished the challenge early ]

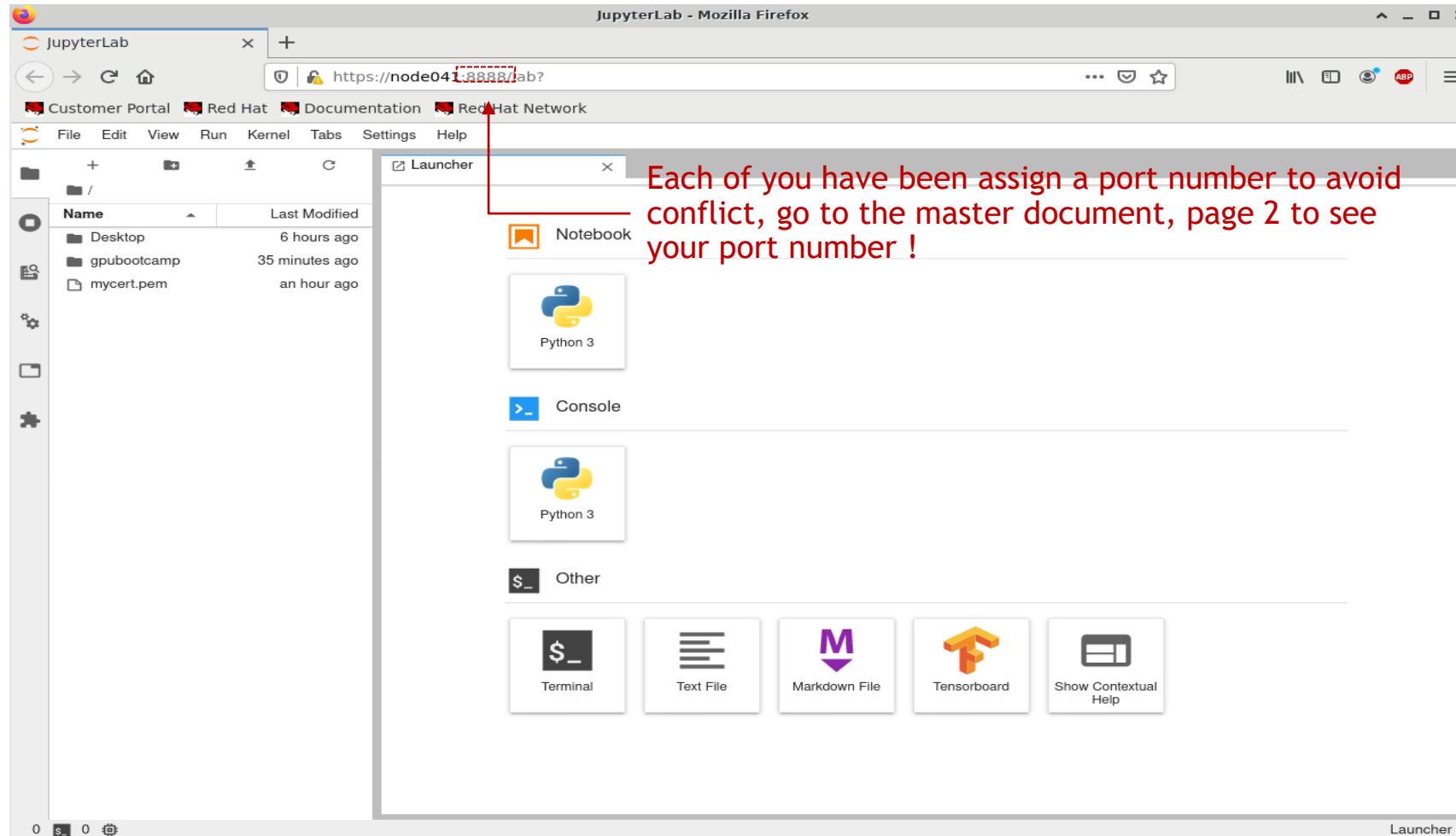
**11:40 AM - 12:00 PM: Discussion & What's up next day**

---

# PREPARE FOR DAY 2'S ENVIRONMENT

Download the instruction from [shared google drive](#) :

Following the step-by-step instruction from that PDF file to access the jupyter lab



**END OF DAY 1**

Congratulations - you did it !



**YOU DID IT!**



# Day 2 - Run through Megatron GPT

## Outline of Day 2 ( 5.5 hours lunch break)

09:00 AM - 09:15 AM: Environment prep

09:15 AM - 09:45 AM: About compute

09:45 AM - 10:15 AM:

Lab 1 - Acquire data ( WebScraping )

Lab 1 - 2\_Estimate Time Needed for Compute

10:15 AM - 10:30 AM: Lab 1 - 3\_Megatron's core MPU (presentation)

10:30 AM - 11:00 AM: Lab 1 - 3\_Megatron's core MPU (lab)

11:00 AM - 11:40 AM:

Lab 1 - 4\_GPT Vocab+Merge file

Lab 1 - 5\_data\_preprocessing2mmap

11:40 AM - 12:00 PM: Intro to profiling

12:00 PM - 12:10 PM: Ask the Experts (Robert Dietrich)

12:10 PM - 13:10 PM: Lunch break

13:10 PM - 13:40 PM: Lab 1 - 6\_GPT training config vs GPUs performance

13:40 PM - 14:10 PM: Challenge overview

14:10 PM - 14:30 PM: Discussion & What's up next day

# INTRODUCING THE SPEAKERS OF THE DAY

Day 2's speakers - Robert Dietrich



**Robert Dietrich**

Expert in Nsight Profiling  
EMEA Profiling Champion



# Day 2 - Run through Megatron GPT

## Outline of Day 2 ( 5.5 hours lunch break)

**09:00 AM - 09:15 AM: Environment prep**

09:15 AM - 09:45 AM: About compute

09:45 AM - 10:15 AM:

Lab 1 - Acquire data ( WebScraping )

Lab 1 - 2\_Estimate Time Needed for Compute

10:15 AM - 10:30 AM: Lab 1 - 3\_Megatron's core MPU (presentation)

10:30 AM - 11:00 AM: Lab 1 - 3\_Megatron's core MPU (lab)

11:00 AM - 11:40 AM:

Lab 1 - 4\_GPT Vocab+Merge file

Lab 1 - 5\_data\_preprocessing2mmap

11:40 AM - 12:00 PM: Intro to profiling

12:00 PM - 12:10 PM: Ask the Experts (Robert Dietrich)

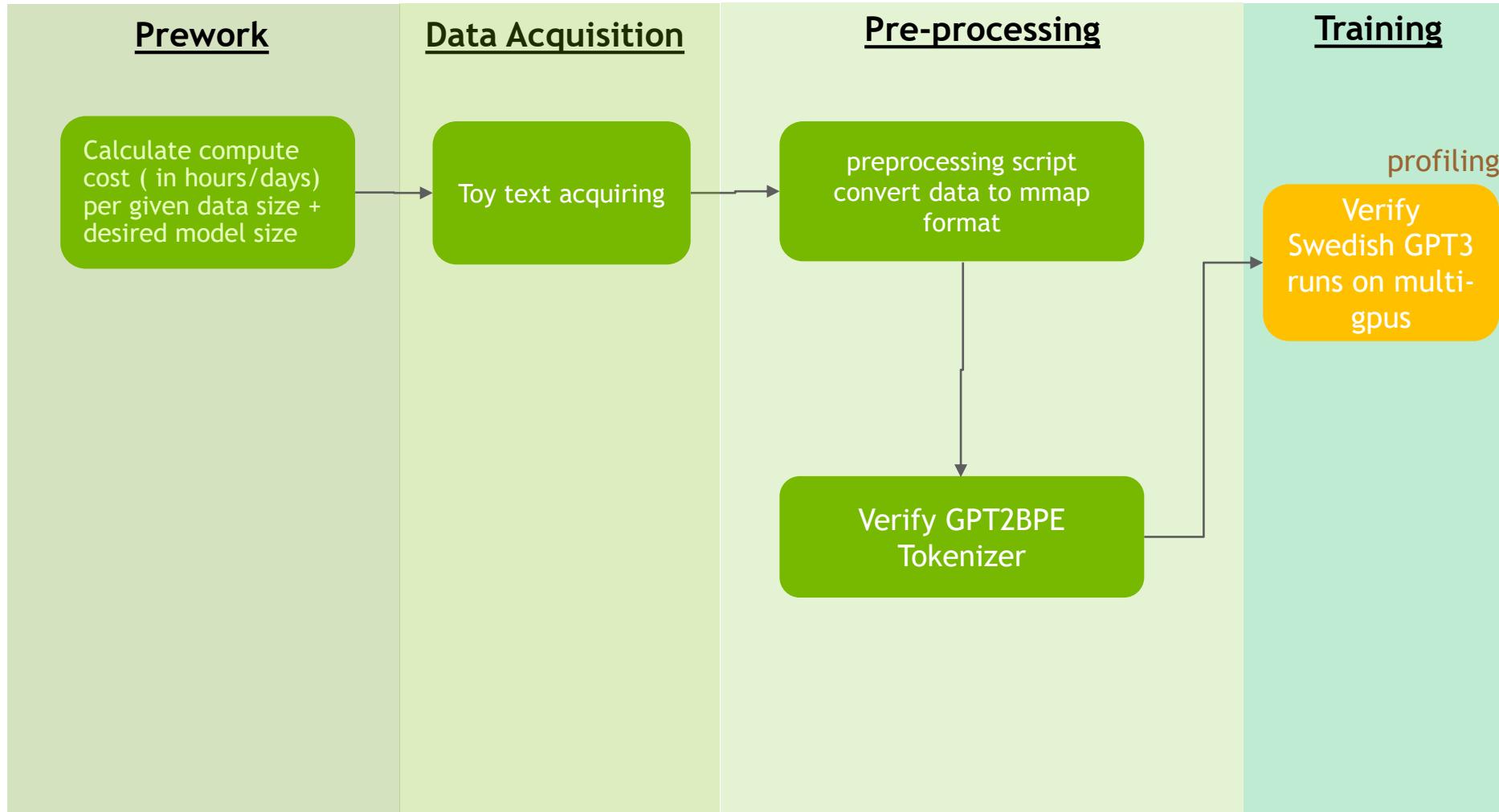
12:10 PM - 13:10 PM: Lunch break

13:10 PM - 13:40 PM: Lab 1 - 6\_GPT training config vs GPUs performance

13:40 PM - 14:10 PM: Challenge overview

14:10 PM - 14:30 PM: Discussion & What's up next day

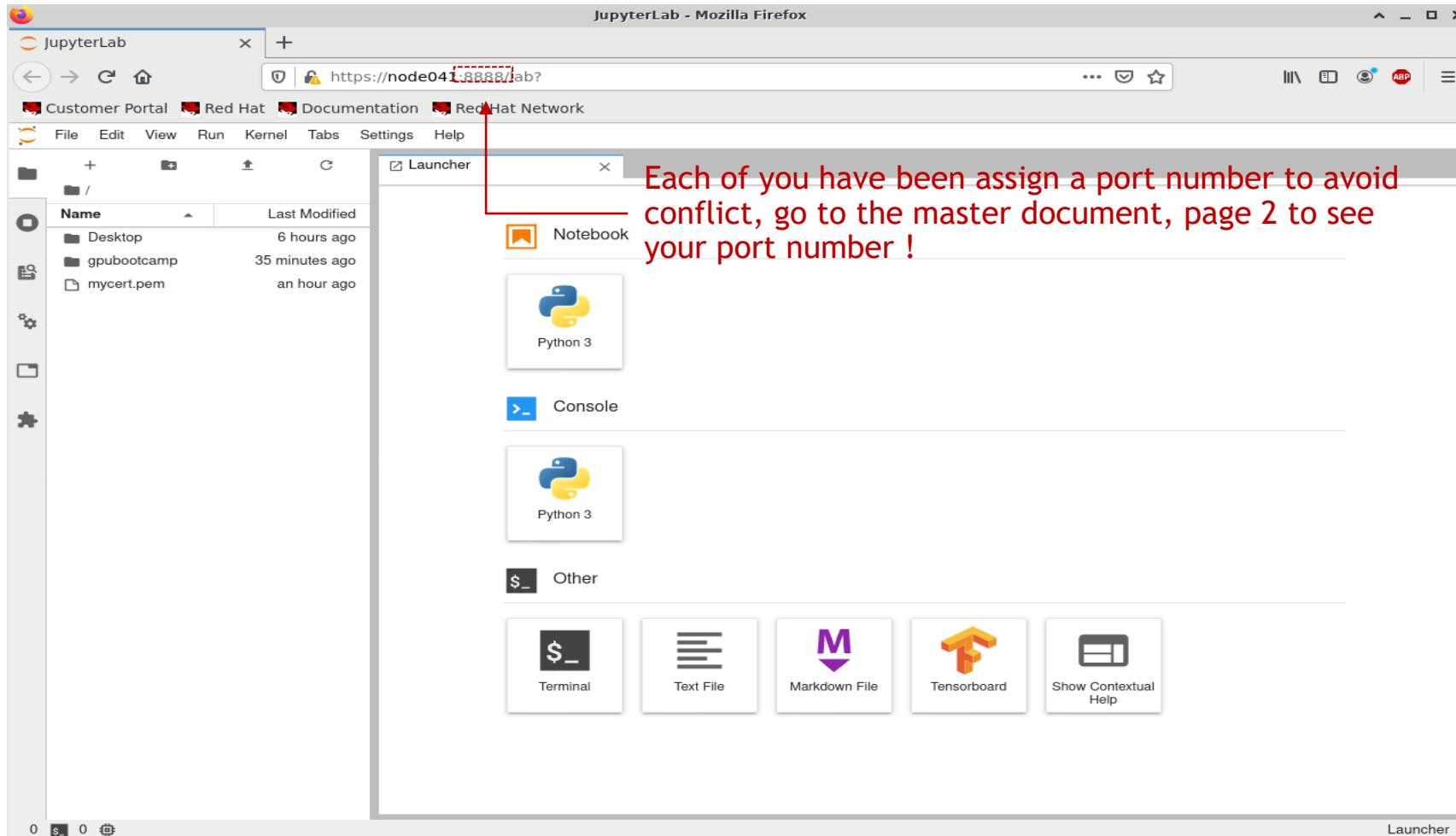
# WORKFLOW OVERVIEW



# DAY 2'S ENVIRONMENT PREPARATION

Download the instruction from [shared google drive](#) :

Following the step-by-step instruction from that PDF file to access the jupyter lab





# Day 2 - Run through Megatron GPT

## Outline of Day 2 ( 5.5 hours lunch break)

09:00 AM - 09:15 AM: Environment prep

**09:15 AM - 09:45 AM: About compute**

09:45 AM - 10:15 AM:

Lab 1 - 1\_Acquire data ( WebScraping )

Lab 1 - 2\_Estimate Time Needed for Compute

10:15 AM - 10:30 AM: Lab 1 - 3\_Megatron's core MPU (presentation)

10:30 AM - 11:00 AM: Lab 1 - 3\_Megatron's core MPU (lab)

11:00 AM - 11:40 AM:

Lab 1 - 4\_GPT Vocab+Merge file

Lab 1 - 5\_data\_preprocessing2mmap

11:40 AM - 12:00 PM: Intro to profiling

12:00 PM - 12:10 PM: Ask the Experts (Robert Dietrich)

12:10 PM - 13:10 PM: Lunch break

13:10 PM - 13:40 PM: Lab 1 - 6\_GPT training config vs GPUs performance

13:40 PM - 14:10 PM: Challenge overview

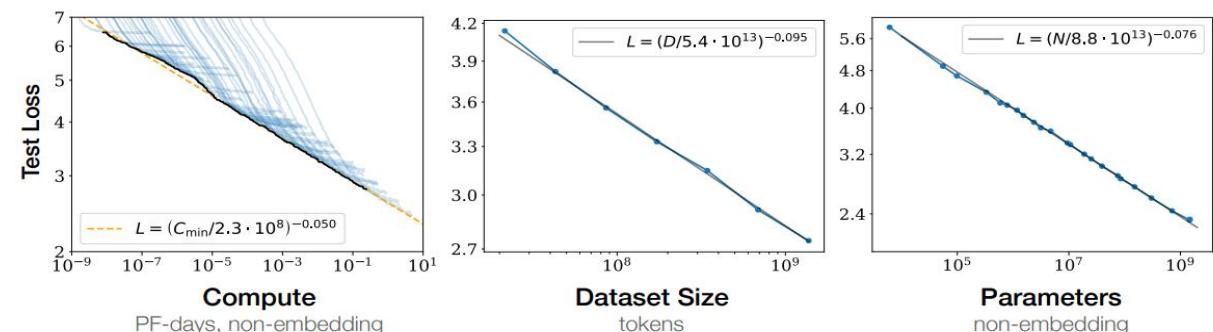
14:10 PM - 14:30 PM: Discussion & What's up next day

# DATA | MODEL SIZE | COMPUTE NEED

## Massive data volume is necessary as you scale

| Model Name            | $n_{\text{params}}$ | $n_{\text{layers}}$ | $d_{\text{model}}$ | $n_{\text{heads}}$ | $d_{\text{head}}$ | Batch Size | Learning Rate        |
|-----------------------|---------------------|---------------------|--------------------|--------------------|-------------------|------------|----------------------|
| GPT-3 Small           | 125M                | 12                  | 768                | 12                 | 64                | 0.5M       | $6.0 \times 10^{-4}$ |
| GPT-3 Medium          | 350M                | 24                  | 1024               | 16                 | 64                | 0.5M       | $3.0 \times 10^{-4}$ |
| GPT-3 Large           | 760M                | 24                  | 1536               | 16                 | 96                | 0.5M       | $2.5 \times 10^{-4}$ |
| GPT-3 XL              | 1.3B                | 24                  | 2048               | 24                 | 128               | 1M         | $2.0 \times 10^{-4}$ |
| GPT-3 2.7B            | 2.7B                | 32                  | 2560               | 32                 | 80                | 1M         | $1.6 \times 10^{-4}$ |
| GPT-3 6.7B            | 6.7B                | 32                  | 4096               | 32                 | 128               | 2M         | $1.2 \times 10^{-4}$ |
| GPT-3 13B             | 13.0B               | 40                  | 5140               | 40                 | 128               | 2M         | $1.0 \times 10^{-4}$ |
| GPT-3 175B or “GPT-3” | 175.0B              | 96                  | 12288              | 96                 | 128               | 3.2M       | $0.6 \times 10^{-4}$ |

**Table 2.1:** Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.



**Figure 1** Language modeling performance improves smoothly as we increase the model size, dataset size, and amount of compute<sup>2</sup> used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

# ESTIMATE COMPUTE NEEDED

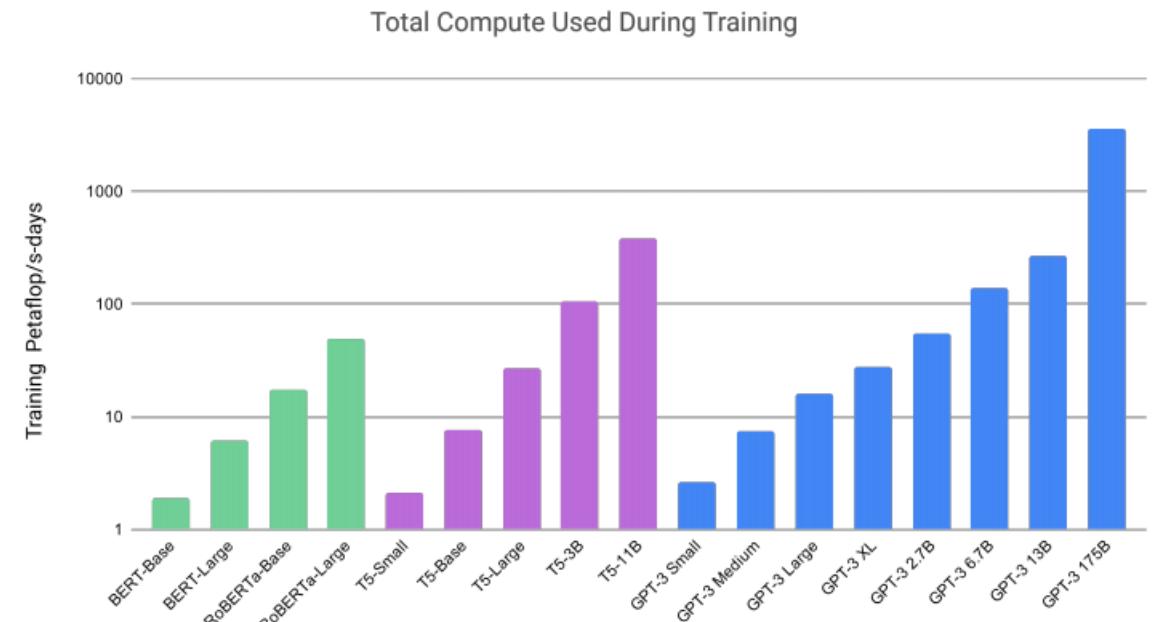
## Calculate how many hours/days compute resource need

paper : <https://arxiv.org/pdf/2005.14165.pdf>

```
[2]: import numpy as np
T=300*1e+9 #oftokens in the dataset
#P=175*1e+9 # number of model parameters
n= 480 # Berzelius 480 # number of GPUs in the compute cluster

def calculate_days_needed(T , P , n ,x):
    if x is None:
        return '1-2 weeks'
    else:
        #x=140*1e+12 # TeraFlop/s per GPU
        tot=8*T*P
        div=n*x
        compute_sec=tot/div
        #convert compute seconds to days
        to_days=round(compute_sec/(3600*24),1)
        return to_days

GPT3_models_labels=[ 'gpt3_2.7B', 'gpt3_6.7B','gpt3_13B', 'gpt3_175B']
GPT3_model_params=[ 2.7*1e+9, 6.7*1e+9 , 13*1e+9, 175*1e+9,1*1e+12 ]
GPT3_model_params_str=['1.3 Billion' , '2.7 Billion', '13 Billion', '175 Billion']
#according to the table above
GPT3_X=[127*1e+12, 130*1e+12,135*1e+12,140*1e+12 ]
print("all below are measured with dataset size **300 billion** measured in tokens \n")
for gpt3_name, gpt3_params, gpt3_param_str, x in zip(GPT3_models_labels,GPT3_model_params,GPT3_model_params_str,GPT3_X):
    days_needed=calculate_days_needed(T,gpt3_params,n,x)
    print("-----")
    print(" language model :{} with {} number of parameters , it will need {} days to compute".format(gpt3_name,gpt3_param_str,days_needed))
    print("-----")
    print(" all below are measured with dataset size **300 billion** measured in tokens\n")
language model :gpt3_2.7B with 1.3 Billion number of parameters , it will need 1.2 days to compute
-----
language model :gpt3_6.7B with 2.7 Billion number of parameters , it will need 3.0 days to compute
-----
language model :gpt3_13B with 13 Billion number of parameters , it will need 5.6 days to compute
-----
language model :gpt3_175B with 175 Billion number of parameters , it will need 72.3 days to compute
```



**Figure 2.2: Total compute used during training.** Based on the analysis in Scaling Laws For Neural Language Models [KMH<sup>+</sup>20] we train much larger models on many fewer tokens than is typical. As a consequence, although GPT-3 3B is almost 10x larger than RoBERTa-Large (355M params), both models took roughly 50 petaflop/s-days of compute during pre-training. Methodology for these calculations can be found in Appendix D.

Source :<https://arxiv.org/pdf/2005.14165.pdf>

# MEGATRON IS AN EFFICIENT FRAMEWORK TO TRAIN VERY LARGE LANGUAGE MODELS

| Model size | Attention heads | Hidden size | Number of layers | Number of parameters (billion) | Model-parallel size | Number of GPUs | Microbatch size | Batch size | Achieved teraFLOP/s per GPU | Percentage of theoretical peak FLOP/s | Achieved aggregate petaFLOP/s |
|------------|-----------------|-------------|------------------|--------------------------------|---------------------|----------------|-----------------|------------|-----------------------------|---------------------------------------|-------------------------------|
| 1.7B       | 24              | 2304        | 24               | 1.7                            | 1                   | 32             | 16              | 512        | 137                         | 44%                                   | 4.4                           |
| 3.6B       | 32              | 3072        | 30               | 3.6                            | 2                   | 64             | 16              | 512        | 138                         | 44%                                   | 8.8                           |
| 7.5B       | 32              | 4096        | 36               | 7.5                            | 4                   | 128            | 16              | 512        | 142                         | 46%                                   | 18.2                          |
| 18B        | 48              | 6144        | 40               | 18.4                           | 8                   | 256            | 8               | 1024       | 135                         | 43%                                   | 34.6                          |
| 39B        | 64              | 8192        | 48               | 39.1                           | 16                  | 512            | 4               | 1536       | 138                         | 44%                                   | 70.8                          |
| 76B        | 80              | 10240       | 60               | 76.1                           | 32                  | 1024           | 2               | 1792       | 140                         | 45%                                   | 143.8                         |
| 145B       | 96              | 12288       | 80               | 145.6                          | 64                  | 1536           | 2               | 2304       | 148                         | 47%                                   | 227.1                         |
| 310B       | 128             | 16384       | 96               | 310.1                          | 128                 | 1920           | 1               | 2160       | 155                         | 50%                                   | 297.4                         |
| 530B       | 128             | 20480       | 105              | 529.6                          | 280                 | 2520           | 1               | 2520       | 163                         | 52%                                   | 410.2                         |
| 1T         | 160             | 25600       | 128              | 1008.0                         | 512                 | 3072           | 1               | 3072       | 163                         | 52%                                   | 502.0                         |

Table 1: Weak scaling throughput for GPT models ranging from 1 billion to 1 trillion parameters



# Day 2 - Your Turn

## Outline of Day 2 ( 5.5 hours lunch break)

09:00 AM - 09:15 AM: Environment prep

09:15 AM - 09:45 AM: About compute

**09:45 AM - 10:15 AM:**

**Lab 1 - 1\_Acquire data ( WebScraping )**

**Lab 1 - 2\_Estimate Time Needed for Compute**

10:15 AM - 10:30 AM: Lab 1 - 3\_Megatron's core MPU (presentation)

10:30 AM - 11:00 AM: Lab 1 - 3\_Megatron's core MPU (lab)

11:00 AM - 11:40 AM:

**Lab 1 - 4\_GPT Vocab+Merge file**

**Lab 1 - 5\_data\_preprocessing2mmap**

11:40 AM - 12:00 PM: Intro to profiling

12:00 PM - 12:10 PM: Ask the Experts (Robert Dietrich)

12:10 PM - 13:10 PM: Lunch break

13:10 PM - 13:40 PM: Lab 1 - 6\_GPT training config vs GPUs performance

13:40 PM - 14:10 PM: Challenge overview

14:10 PM - 14:30 PM: Discussion & What's up next day

# FETCH THE TOY DATA

## Lab1-1\_Website\_scraping.ipynb.ipynb

### Get Your Own Data via webcrawling the website / webpages you have permission to use

note\_1 : strongly recommend to consult with your own legal department for compliance before proceeding this step on websites/webpages you have permission to

note\_2 : modification needed when applying to different website/webpages

note\_3 : use at your own risk !

### Learning Objectives

The goal of this lab is to demonstrate the fact that, there are ways to obtain your own data, the example given below is to crawl the webpages, from a seeding url, with the end goal of obtaining the raw texts. Please double check you have permission to extract the content of these webpages.

- Provide a list of urls in a text file via collecting webpages links

the base url used to crawl links from is <https://developer.nvidia.com/blog/>

- scrap the webpage ( in html format ) using [scrapy](#) and obtain raw text, then concatenate all the raw text per webpage into one text file

```
In [ ]: !pip install beautifulsoup4  
!pip install html5lib  
!pip install PyPDF2  
!pip install selenium  
!pip install Scrapy
```

### crawl NVblog landing page and obtain links to the individual blogs

source github repo : <https://github.com/x4nth055/pythoncode-tutorials/tree/master/web-scraping/link-extractor>

```
In [2]: !wget https://raw.githubusercontent.com/x4nth055/pythoncode-tutorials/master/web-scraping/link-extractor/link_extractor.py  
--2021-09-15 09:14:55-- https://raw.githubusercontent.com/x4nth055/pythoncode-tutorials/master/web-scraping/link-extractor/link_extractor.py  
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.108.133, 185.199.109.133, ...  
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 3485 (3.4K) [text/plain]  
Saving to: 'link_extractor.py'  
  
link_extractor.py 100%[=====] 3.40K --.-KB/s in 0.001s  
2021-09-15 09:14:56 (3.97 MB/s) - 'link_extractor.py' saved [3485/3485]
```



# Day 2 - Intro to Megatron MPU



Xianchao Wu

## Outline of Day 2 ( 5.5 hours lunch break)

09:00 AM - 09:15 AM: Environment prep

09:15 AM - 09:45 AM: About compute

09:45 AM - 10:15 AM:

Lab 1 - Acquire data ( WebScraping )

Lab 1 - 2\_Estimate Time Needed for Compute

**10:15 AM - 10:30 AM: Lab 1 - 3\_Megatron's core MPU (presentation)**

10:30 AM - 11:00 AM: Lab 1 - 3\_Megatron's core MPU (lab)

11:00 AM - 11:40 AM:

Lab 1 - 4\_GPT Vocab+Merge file

Lab 1 - 5\_data\_preprocessing2mmap

11:40 AM - 12:00 PM: Intro to profiling

12:00 PM - 12:10 PM: Ask the Experts (Robert Dietrich)

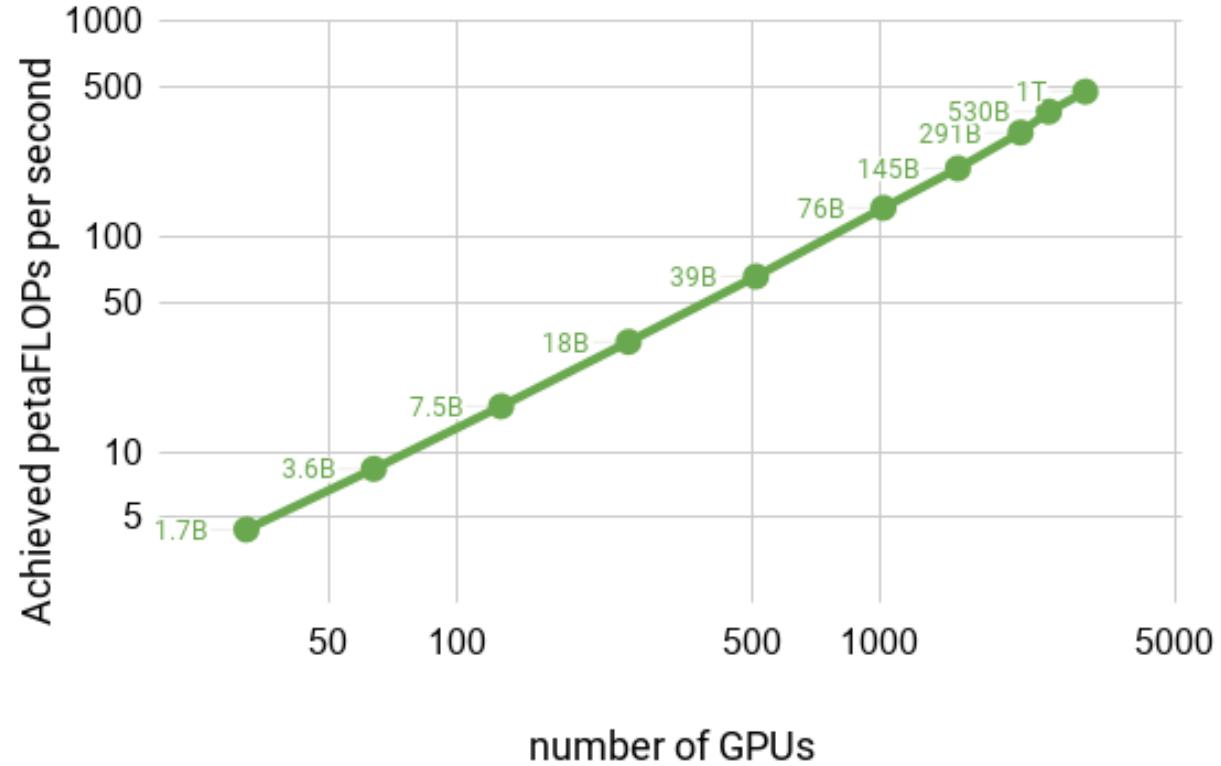
12:10 PM - 13:10 PM: Lunch break

13:10 PM - 13:40 PM: Lab 1 - 6\_GPT training config vs GPUs performance

13:40 PM - 14:10 PM: Challenge overview

14:10 PM - 14:30 PM: Discussion & What's up next day

# REMEMBER WHY WE CARE



● is the achievable GPT3 model size

# MEGATRON CORE MPU

NVIDIA / Megatron-LM

Watch 85 Star 2.3k Fork 442

Code Issues 75 Pull requests 6 Actions Security Insights

main ▾ Megatron-LM / megatron / mpu / Go to file Add file ...

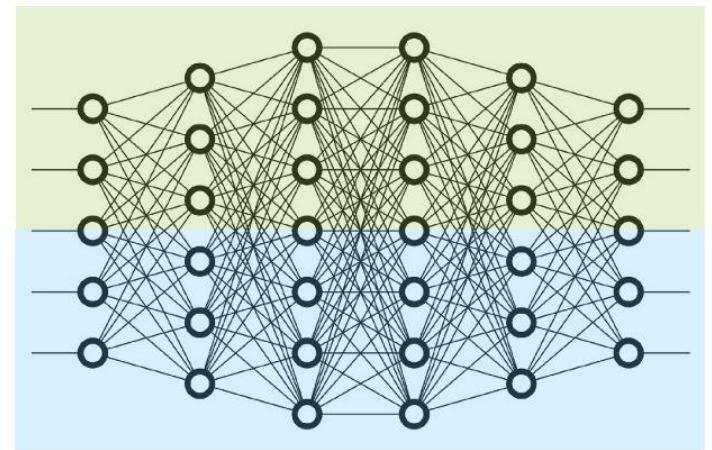
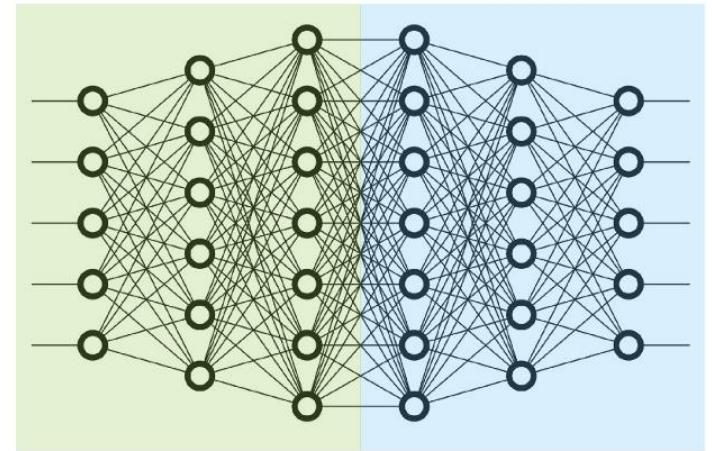
mshoeybi only support pp=1 7b58544 19 days ago History

..

| File             | Description  | Time Ago      |
|------------------|--|---------------|
| tests            | Intra-layer MP -> Tensor MP, Inter-layer MP -> Pipeline MP | 10 months ago |
| __init__.py      | removed contiguous buffer for checkpointed activation      | 19 days ago   |
| cross_entropy.py | Intra-layer MP -> Tensor MP, Inter-layer MP -> Pipeline MP | 10 months ago |
| data.py          | vision transformer model and vision classification task    | 8 months ago  |
| initialize.py    | Destroy more groups in <code>destroy_model_parallel</code> | 21 days ago   |
| layers.py        | fix typo   | 2 months ago  |
| mappings.py      | fix typo in mappings.py                                    | 2 months ago  |
| random.py        | only support pp=1  | 19 days ago   |
| utils.py         | changed licence 2019 to 2020                               | 17 months ago |

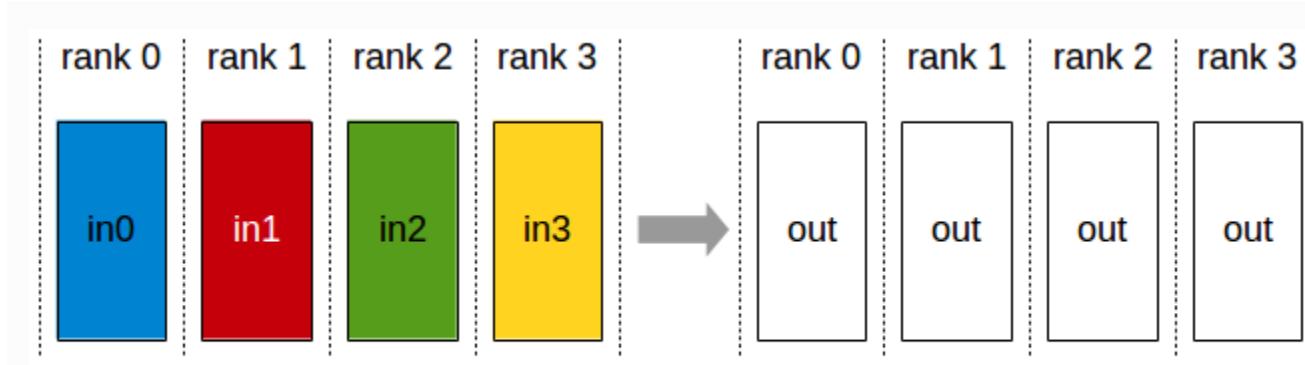
# MODEL PARALLELISM

- Pipeline (Inter-Layer) Parallelism
  - Split sets of layers across multiple devices
  - Layer 0,1,2 and layer 3,4,5 are on different devices
- Tensor (Intra-Layer) Parallelism
  - Split individual layers across multiple devices
  - Both devices compute different parts of Layer 0,1,2,3,4,5



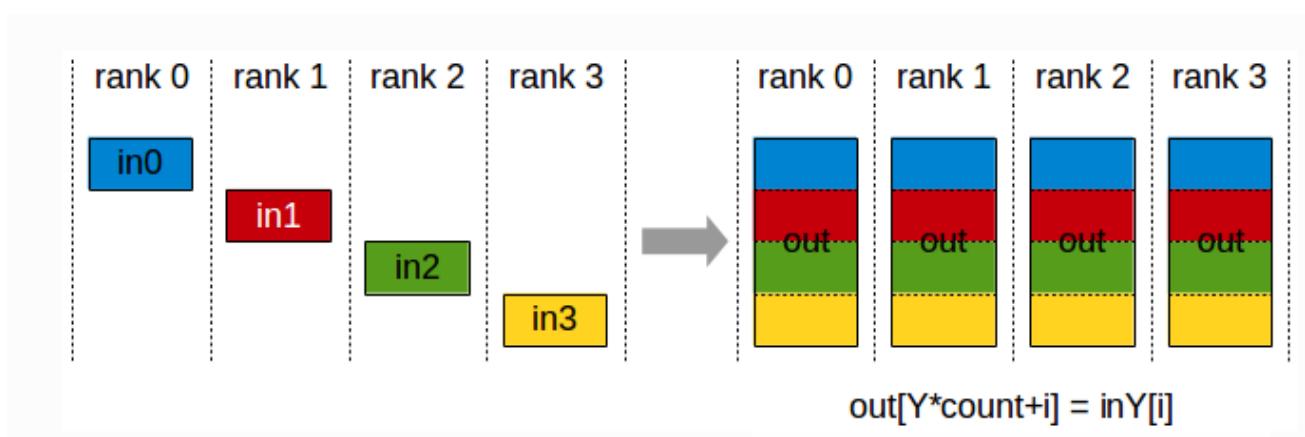
# NCCL COLLECTIVE OPERATORS

All reduce



*All-Reduce operation: each rank receives the reduction of input values across ranks.*

All gather



*AllGather operation: each rank receives the aggregation of data from all ranks in the order of the ranks.*

# GELU | ELU | RELU

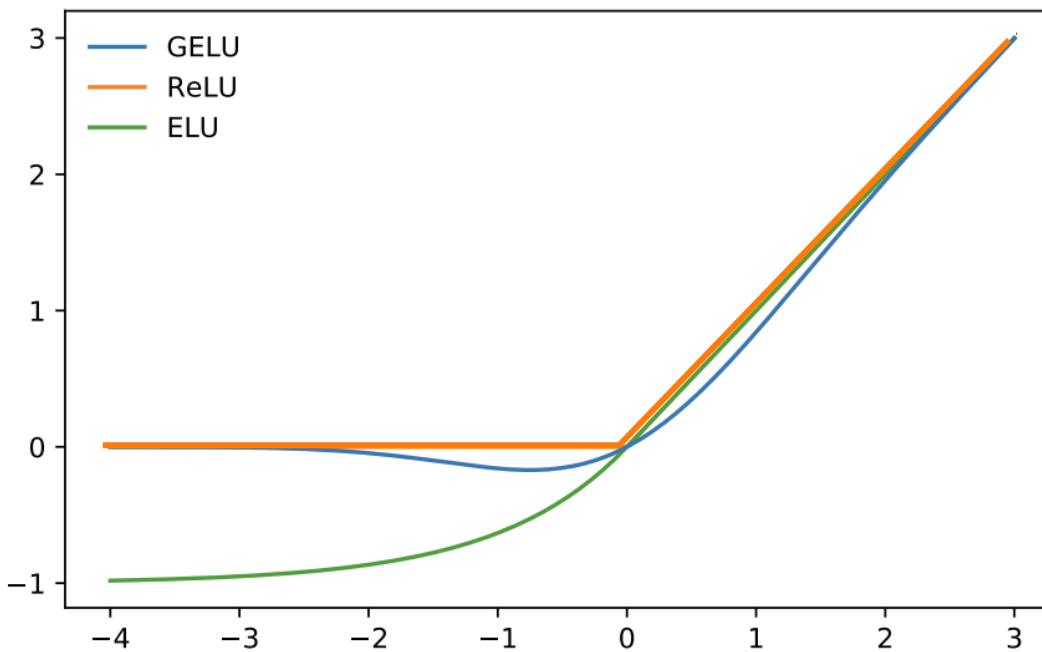


Figure 1: The GELU ( $\mu = 0, \sigma = 1$ ), ReLU, and ELU ( $\alpha = 1$ ).

# DROPOUT

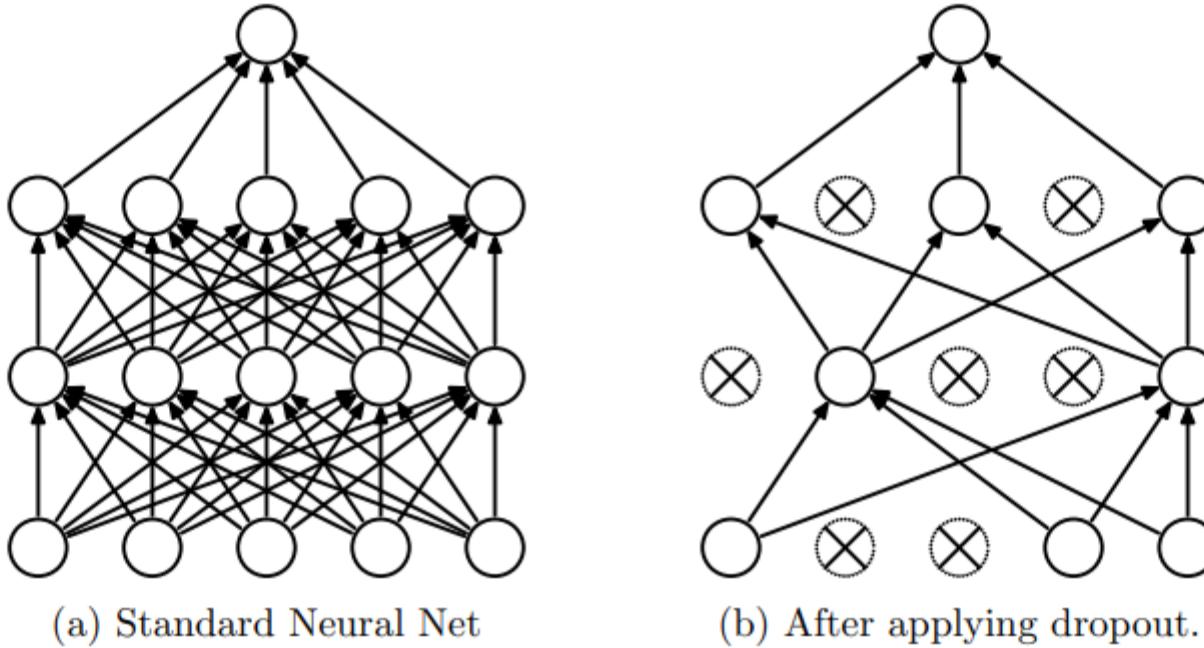
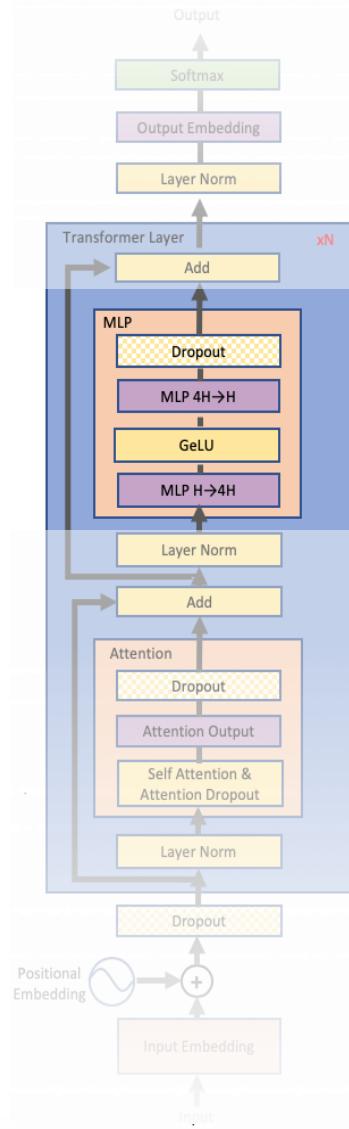


Figure 1: Dropout Neural Net Model. **Left:** A standard neural net with 2 hidden layers. **Right:** An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

# LET'S BREAK IT DOWN - FIRST UP *MLP*



# PARTITIONING MULTI-LAYER PERCEPTRON (MLP)

- MLP:

$$Y = \text{GeLU}(XA)$$

$$Z = \text{Dropout}(YB)$$

- Approach 1: split X column-wise and A row-wise:

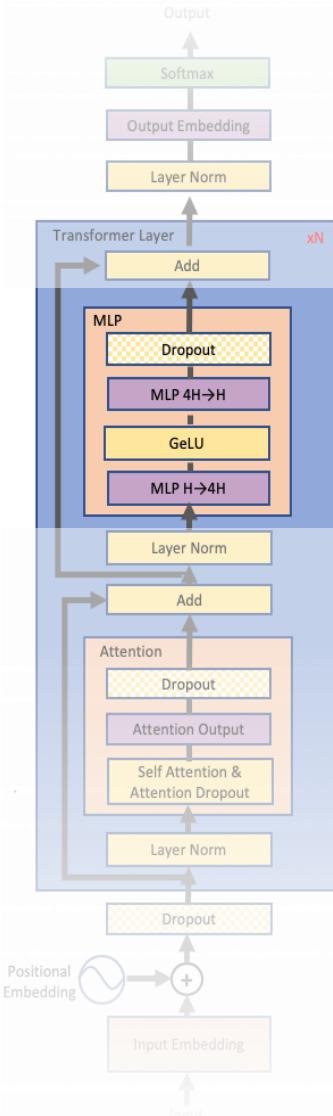
$$X = [X_1, X_2] \quad A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \longrightarrow Y = \text{GeLU}(X_1A_1 + X_2A_2)$$

- Before GeLU we will need a communication point

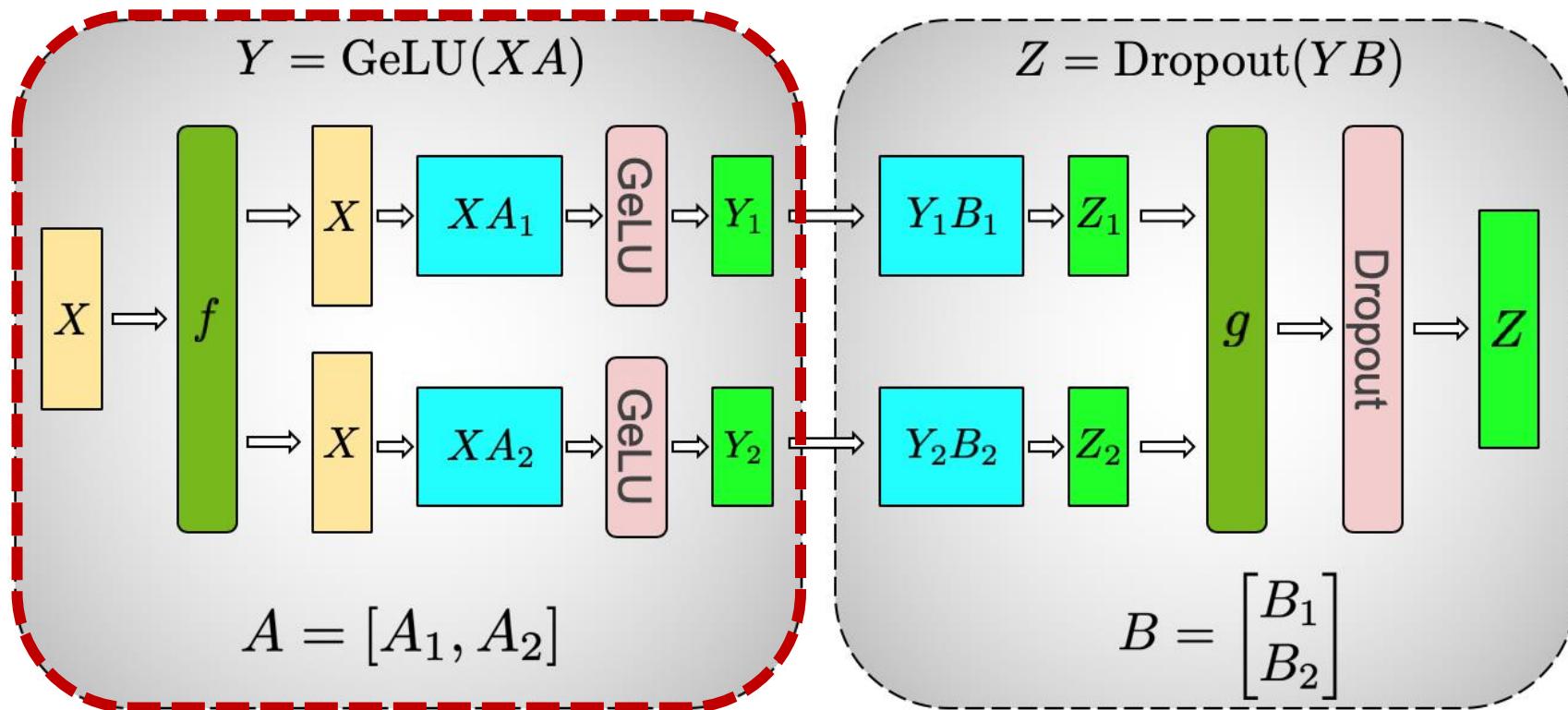
- Approach 2: split A column-wise:

$$A = [A_1, A_2] \longrightarrow [Y_1, Y_2] = [\text{GeLU}(XA_1), \text{GeLU}(XA_2)]$$

- No communication is required



# MLP



$f$  and  $g$  are conjugate,  $f$  is identity operator in the forward pass and all-reduce in the backward pass while  $g$  is all-reduce in forward and identity in backward.

# TENSOR MODEL PARALLEL - COLUMN PARALLEL

$$Y = \text{gelu}(XA)$$

If we cut A vertically

$$A = [A_1, A_2]$$

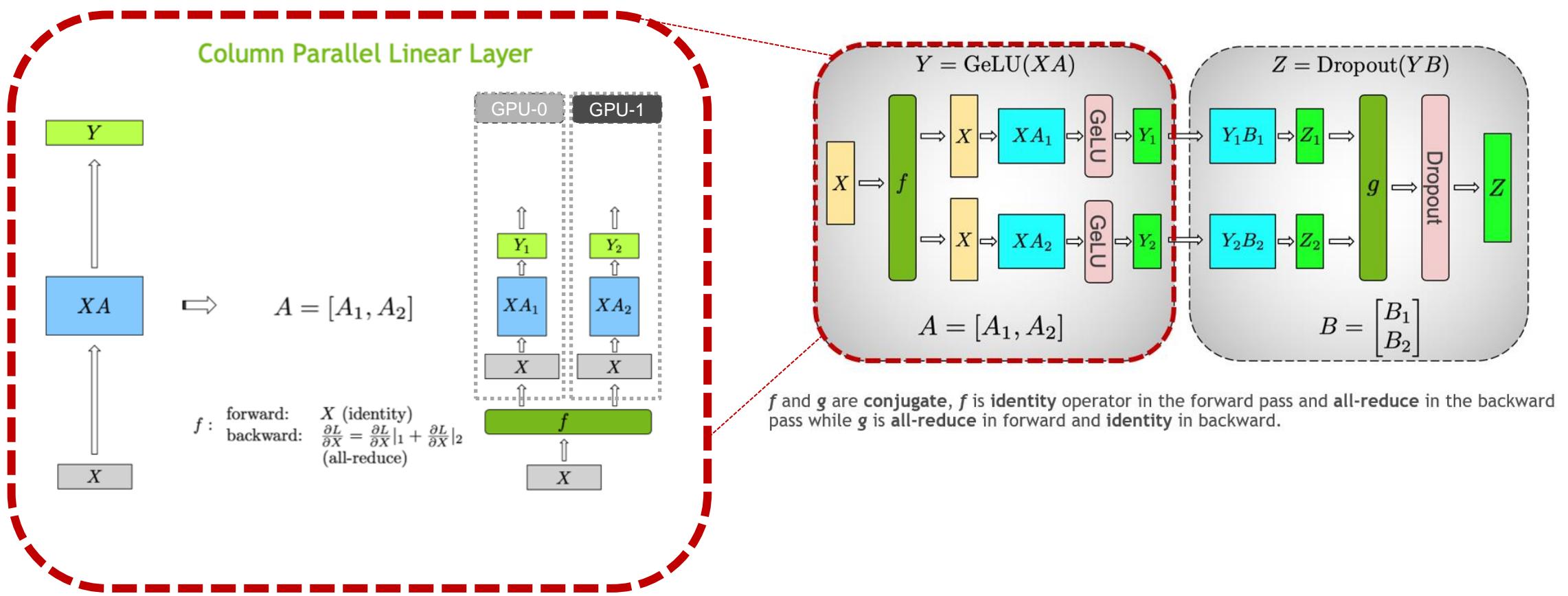
| slice           | Parameter matrix A | Input tensor X |
|-----------------|--------------------|----------------|
| Column Parallel | $[A_1, A_2]$       | X              |

$$Y = [Y_1, Y_2] = [\text{gelu}(XA_1), \text{gelu}(XA_2)]$$

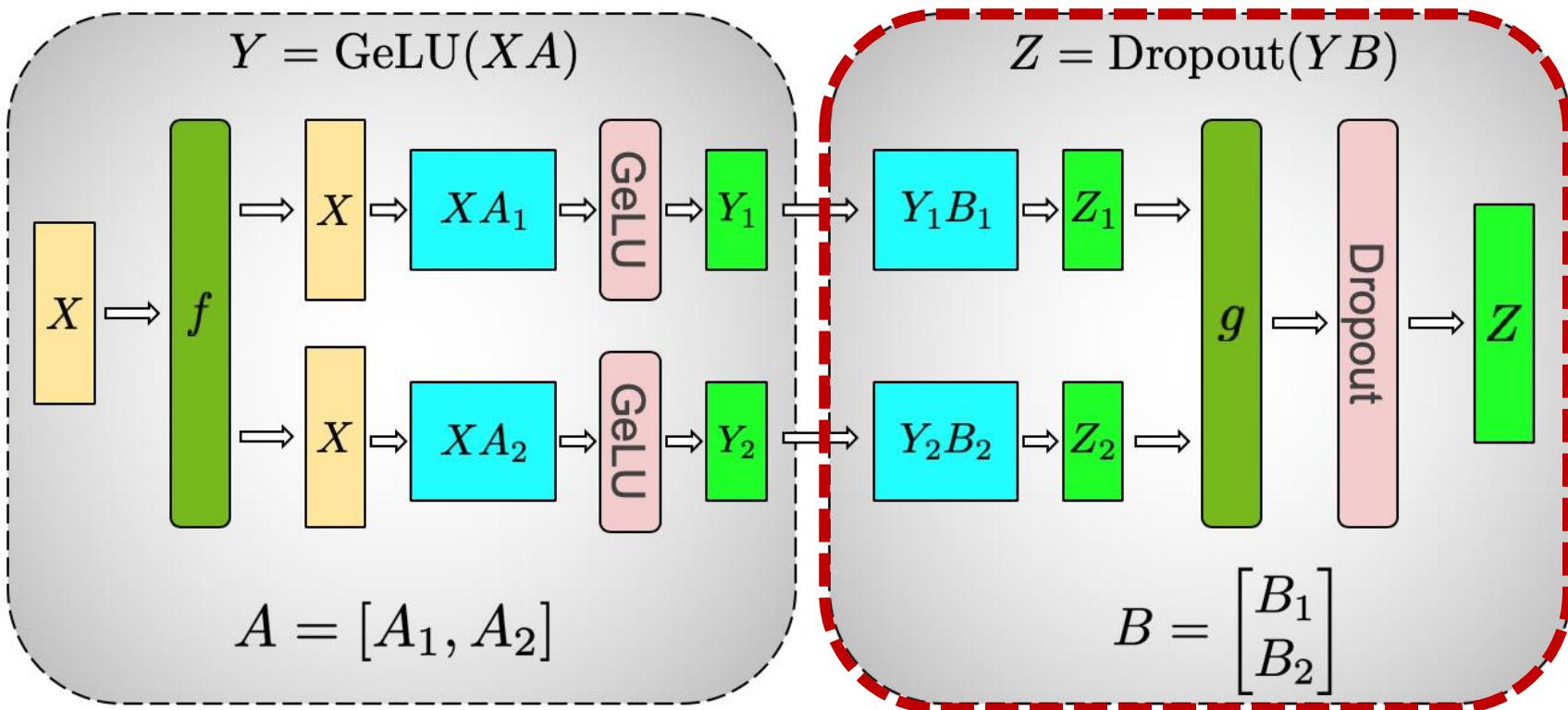
GPU-0

GPU-1

# COLUMN PARALLEL - FORWARD | BACKWARD



# MLP



$f$  and  $g$  are conjugate,  $f$  is identity operator in the forward pass and all-reduce in the backward pass while  $g$  is all-reduce in forward and identity in backward.

# TENSOR MODEL PARALLEL - ROW PARALLEL

Z=Dropout(YB)

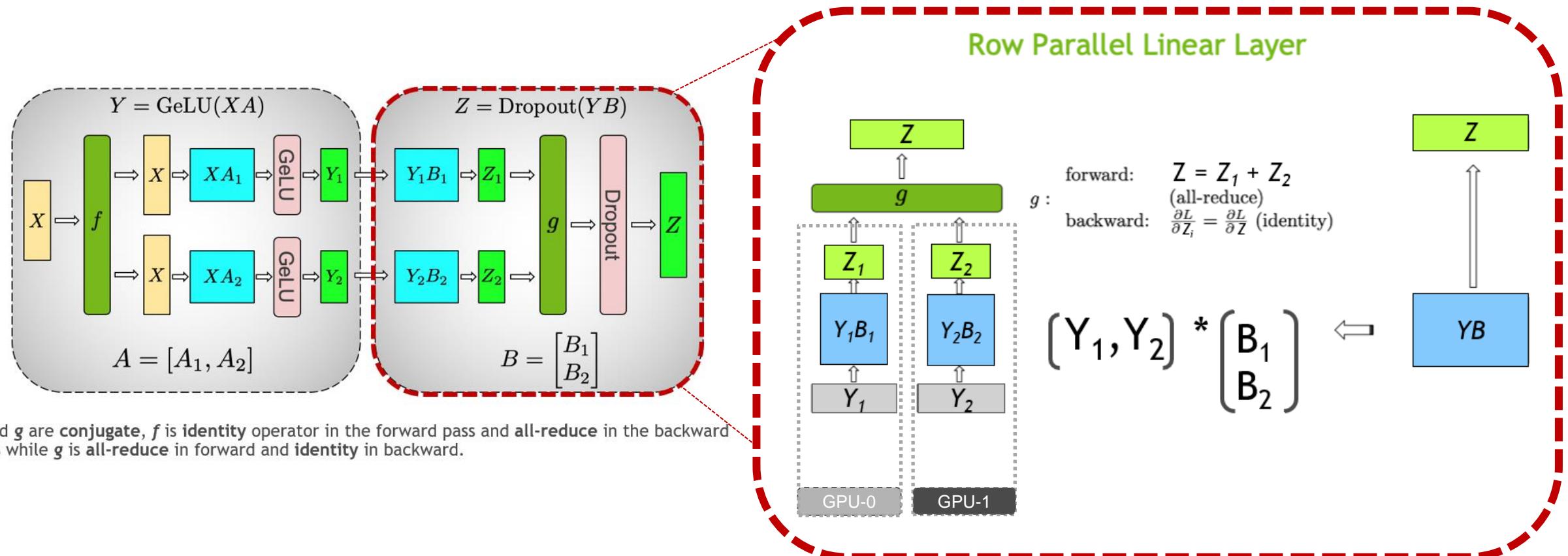
If we cut B horizontally

| slice        | Parameter Matrix B | output from previous Y |
|--------------|--------------------|------------------------|
| Row Parallel | $[B_1, B_2]$       | $[Y_1, Y_2]$           |

$$\begin{array}{l} \text{GPU-0} \\ \text{GPU-1} \end{array} \Rightarrow \begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} = [Y_1, Y_2] * \begin{pmatrix} B_1 \\ B_2 \end{pmatrix}$$

$$Z = Z_1 + Z_2$$

# ROW PARALLEL - FORWARD | BACKWARD



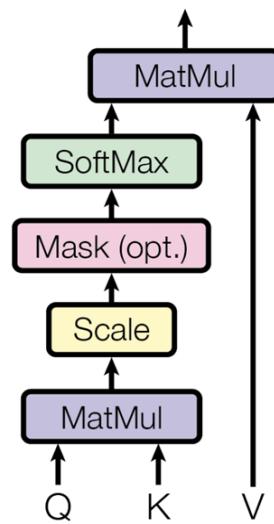
A network graph visualization on a dark gray background. The graph consists of numerous small, semi-transparent circular nodes. Some nodes are white, while others are a bright lime green. These nodes are interconnected by a dense web of thin, light gray lines representing edges. The overall effect is a complex, organic, and abstract representation of a network structure.

SELF-ATTENTION

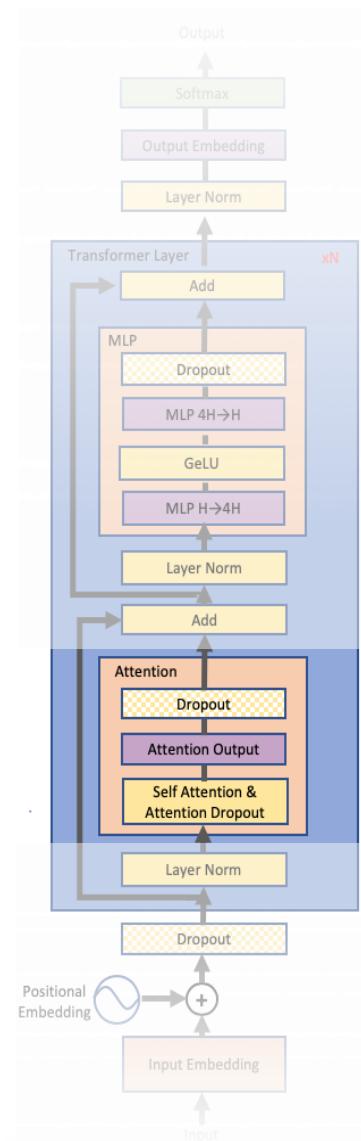
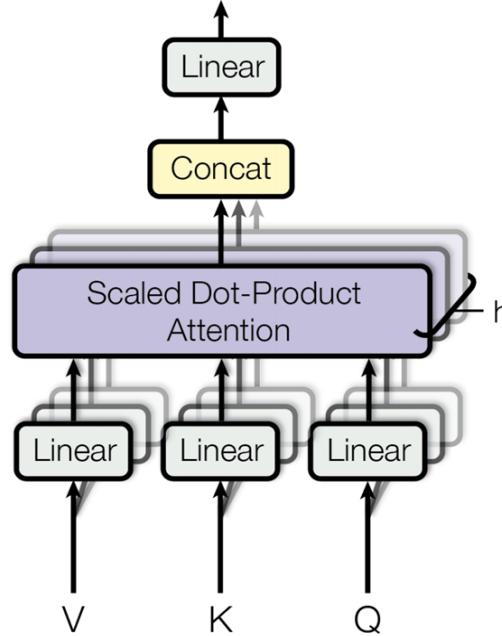
# PARTITIONING: SELF-ATTENTION

- Self-attention is more complex than MLP

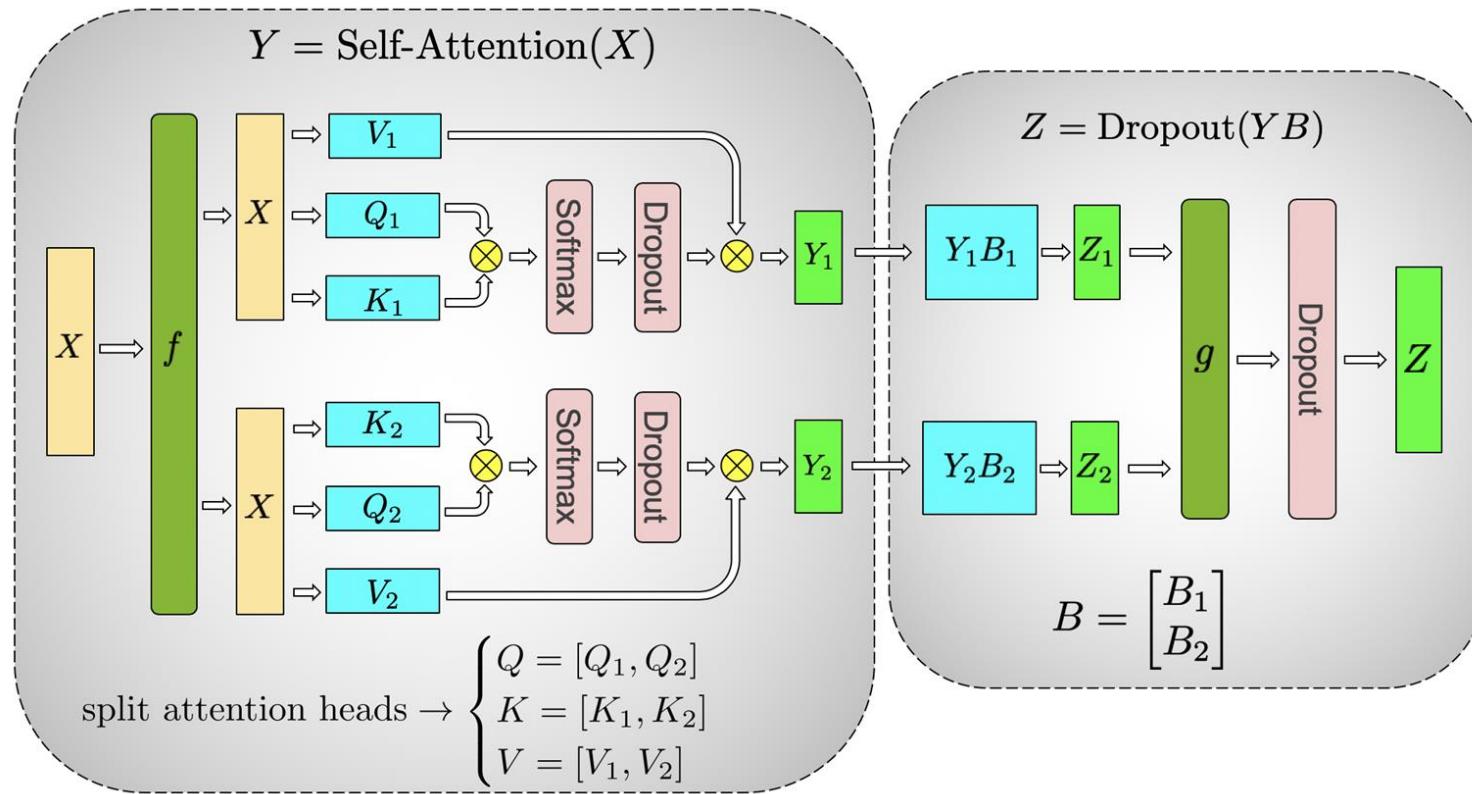
Scaled Dot-Product Attention



Multi-Head Attention



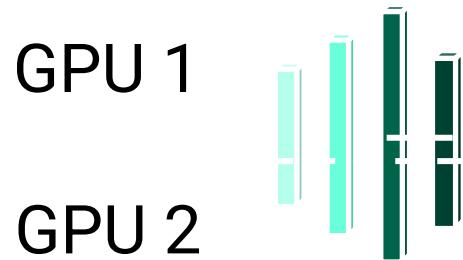
# SELF-ATTENTION - SAME MECHANISM APPLIES



$f$  and  $g$  are conjugate,  $f$  is identity operator in the forward pass and all-reduce in the backward pass while  $g$  is all-reduce in forward and identity in backward.

# COMPARING TENSOR AND PIPELINE PARALLELISM

## Tensor Parallelism



Communication expensive

Good performance across  
batch sizes

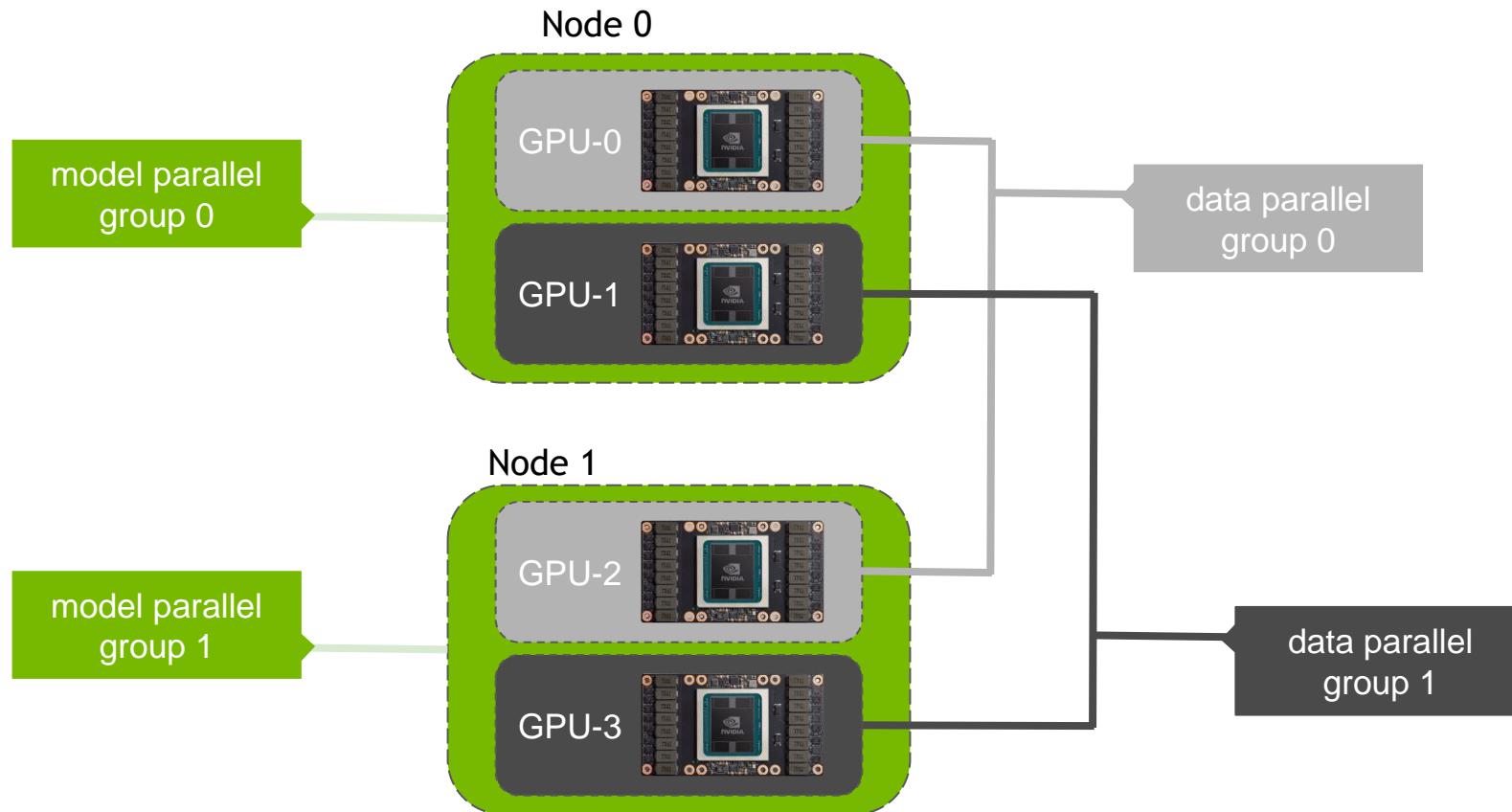
## Pipeline Parallelism



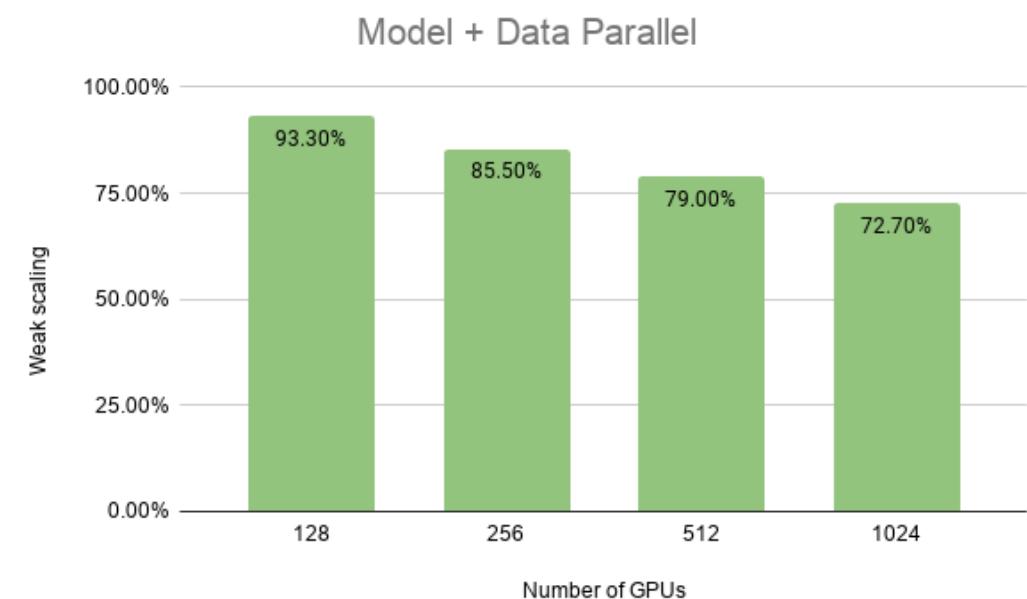
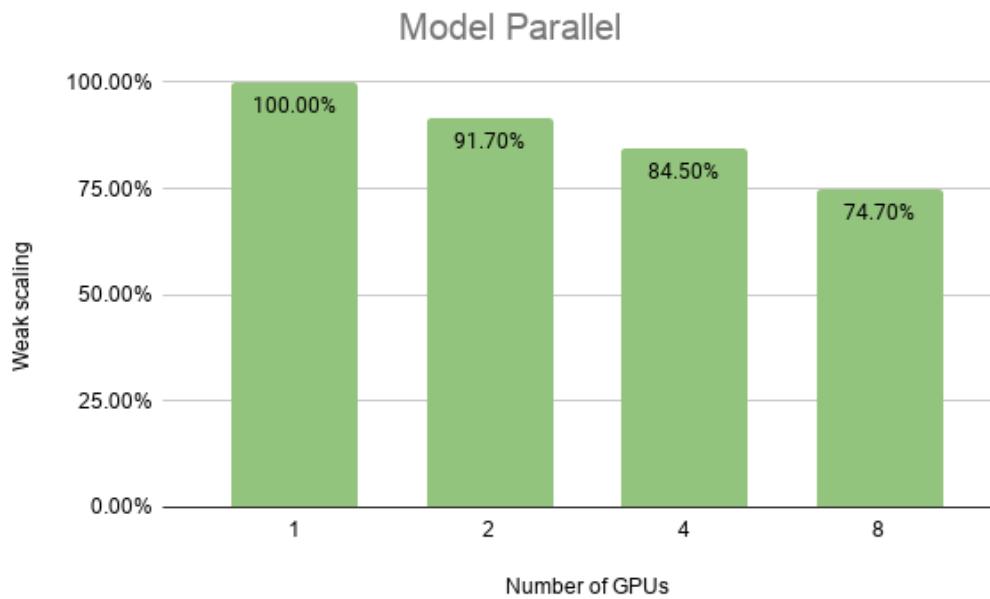
Communication cheap

Good performance at  
larger batch sizes  
(pipeline stall amortized)

# HYBRID MODEL+DATA PARALLELISM



# WEAK SCALING EFFICIENCY ON SELENE

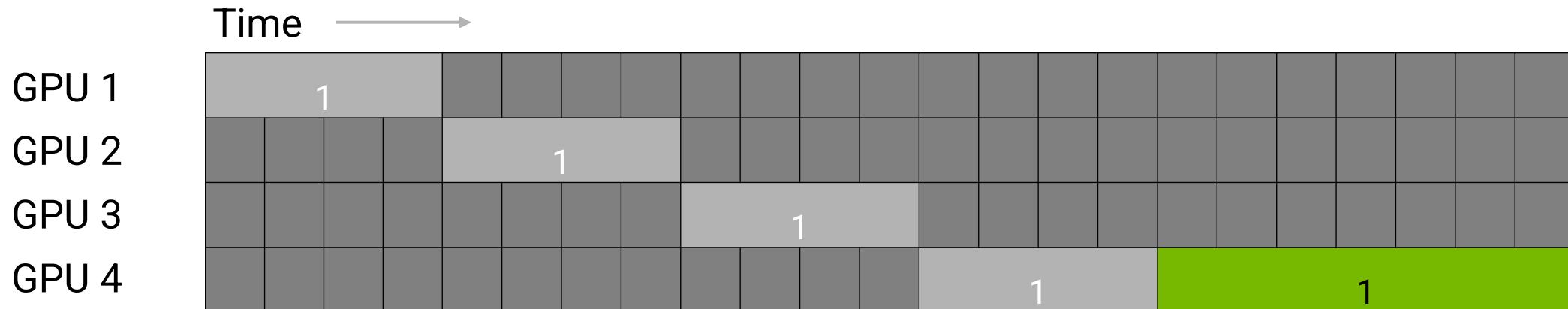


Baseline (1.2B parameters on a single GPU) sustains  
149.8 teraFLOPs/sec on a single A100 GPU (48% of the theoretical peak flops) over the entire training process

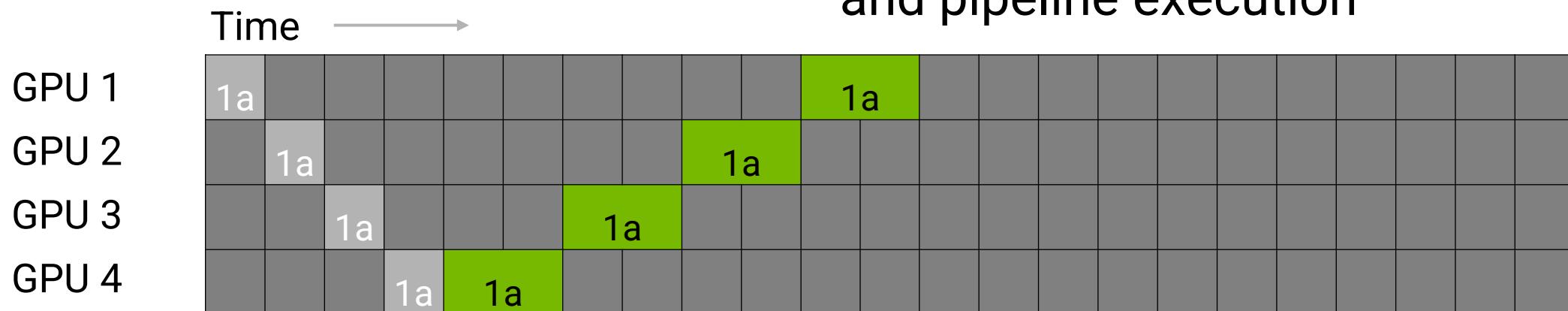


PIPELINING

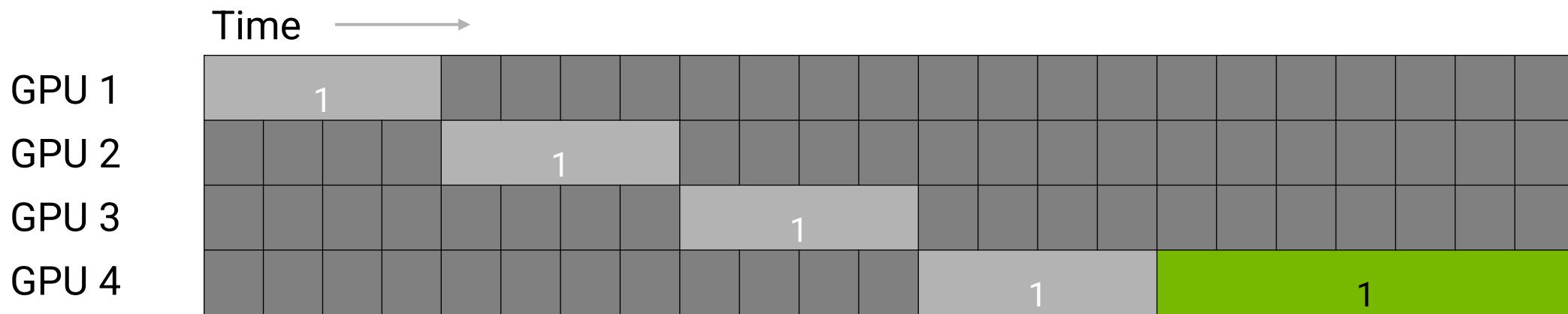
# PIPELINING



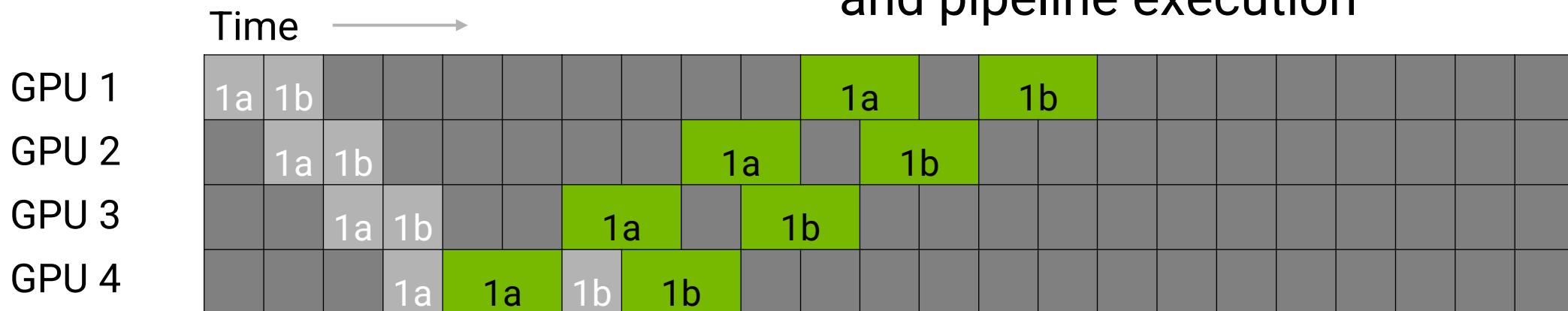
# Split batch into microbatches and pipeline execution



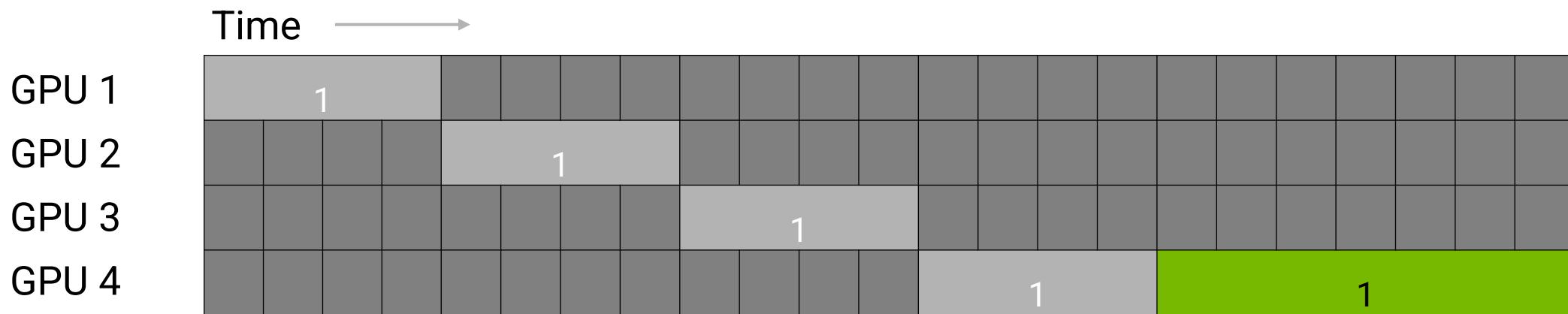
# PIPELINING



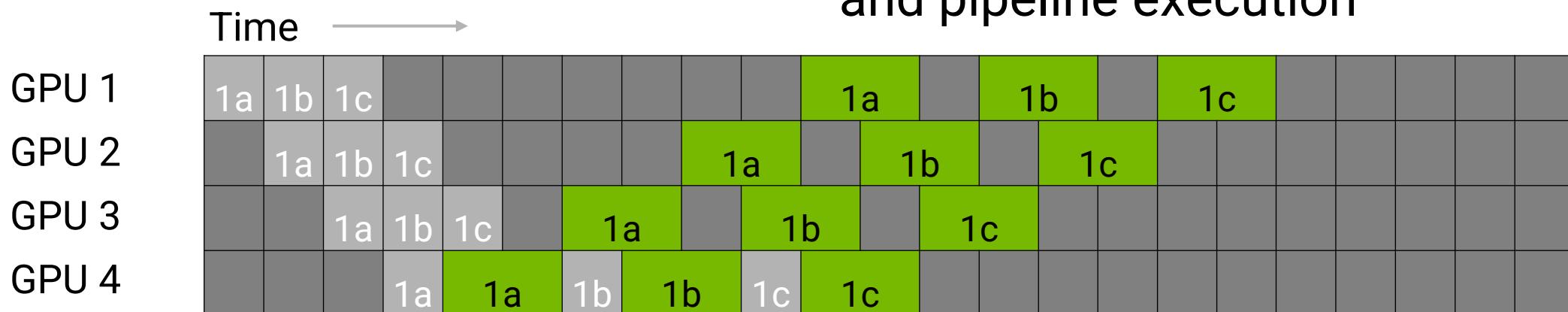
# Split batch into microbatches and pipeline execution



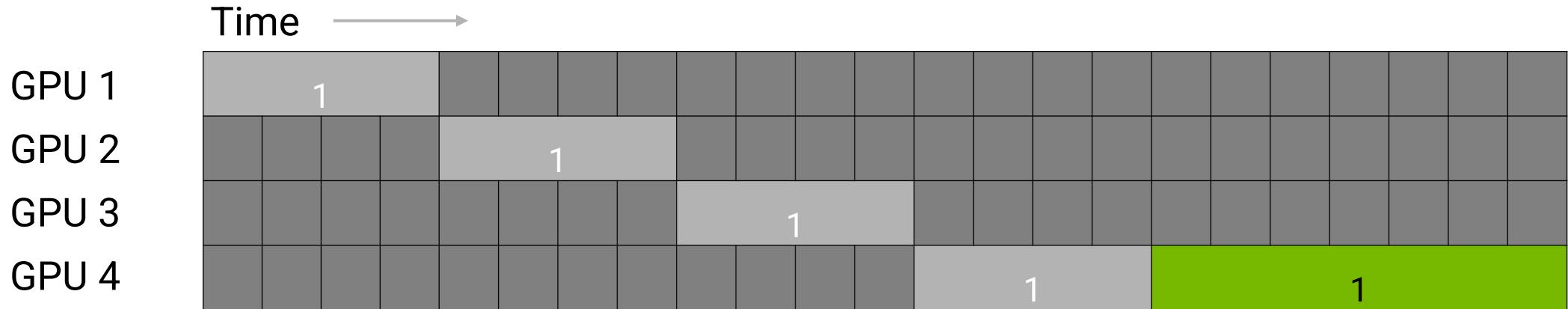
# PIPELINING



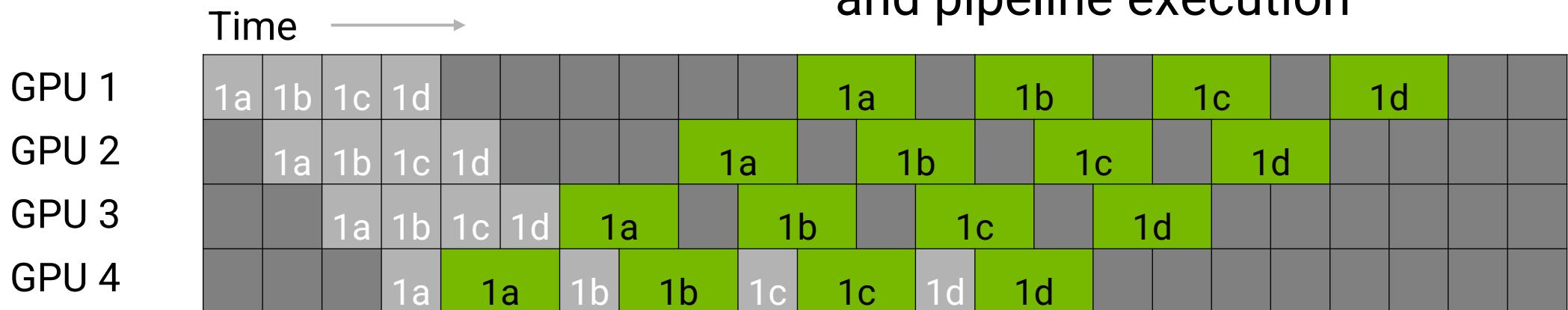
# Split batch into microbatches and pipeline execution



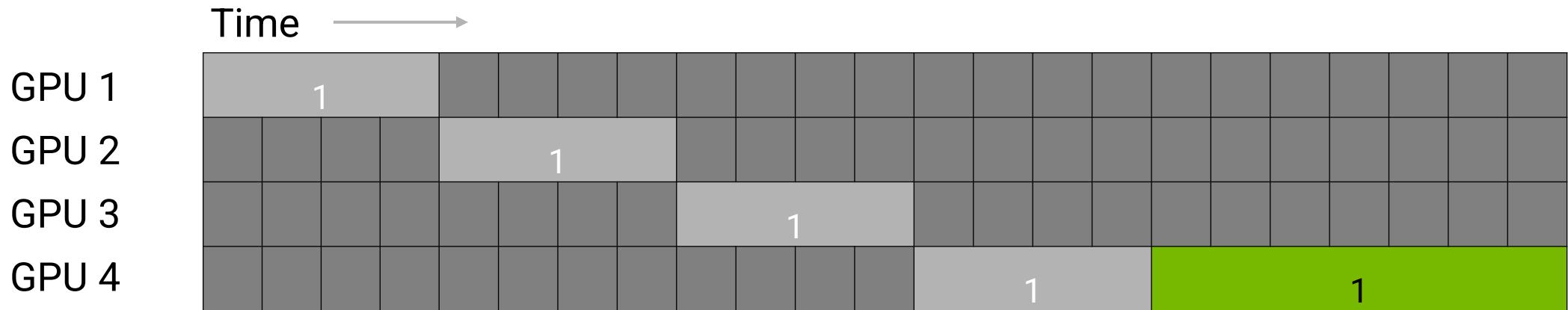
# PIPELINING



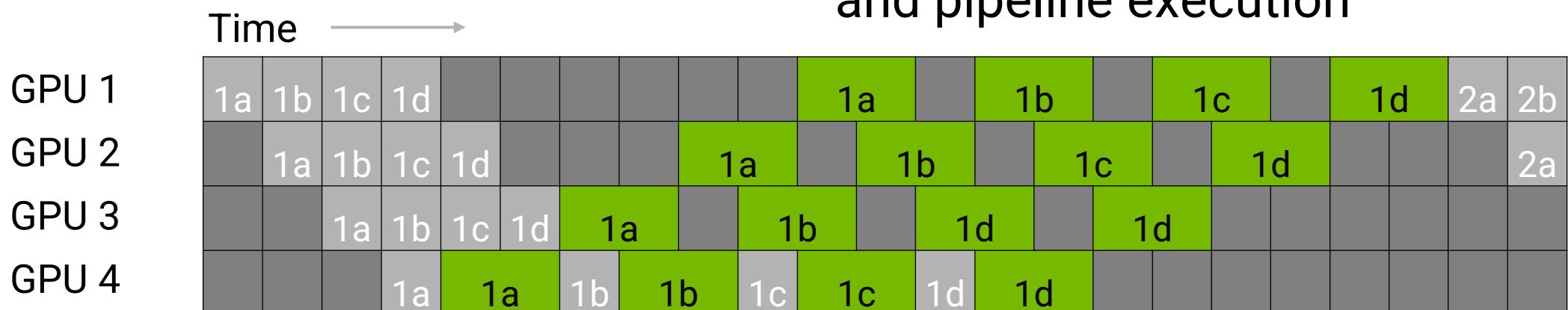
Split batch into microbatches  
and pipeline execution



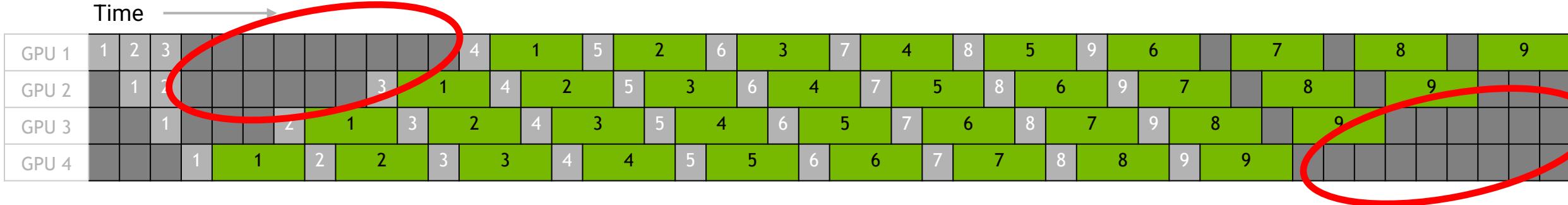
# PIPELINING



Split batch into microbatches  
and pipeline execution



# PIPELINE BUBBLES



$p$  : number of pipeline stages

$m$  : number of micro batches

$t_f$  : forward step time

$t_b$  : backward step time



$$\text{total time} = (m + p - 1) \times (t_f + t_b)$$

$$\text{ideal time} = m \times (t_f + t_b)$$

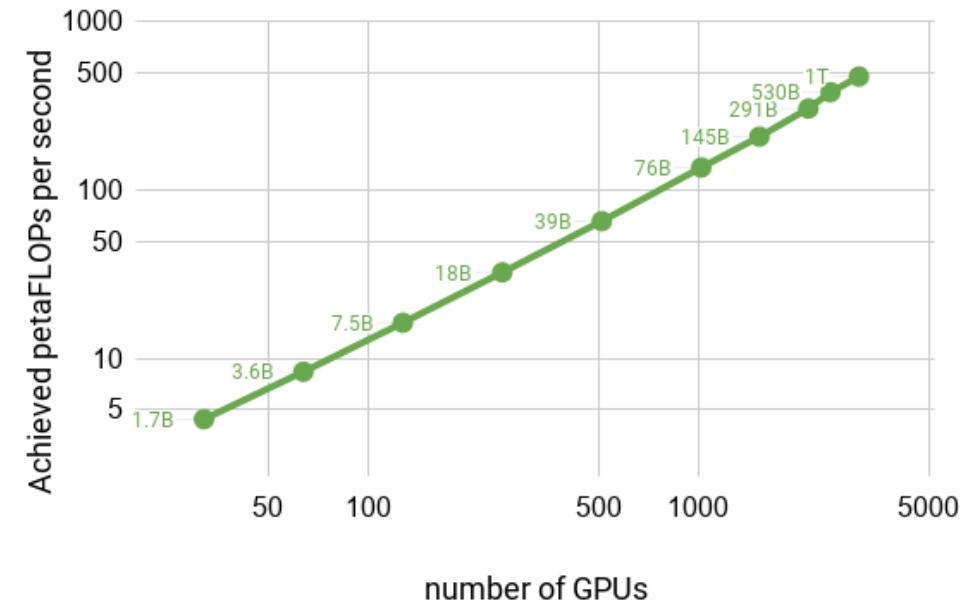
$$\text{bubble time} = (p - 1) \times (t_f + t_b)$$

$$\text{bubble time overhead} = \frac{\text{bubble time}}{\text{ideal time}} = \frac{p - 1}{m}$$

# SCALING ON SELENE ( NVIDIA SUPERPOD )

Sequence length: 2048

Vocabulary size: 51,200



| Model size (parameters) | Attention heads | Hidden size | Number layers | Tensor parallel size | Pipeline parallel size | Model parallel size | Data parallel size | Number GPUs | Batch size | % peak flops |
|-------------------------|-----------------|-------------|---------------|----------------------|------------------------|---------------------|--------------------|-------------|------------|--------------|
| 1.7B                    | 24              | 2304        | 24            | 1                    | 1                      | 1                   | 32                 | 32          | 512        | 44%          |
| 3.6B                    | 32              | 3072        | 30            | 2                    | 1                      | 2                   | 32                 | 64          | 512        | 42%          |
| 7.5B                    | 32              | 4096        | 36            | 4                    | 1                      | 4                   | 32                 | 128         | 512        | 41%          |
| 18B                     | 48              | 6144        | 40            | 8                    | 1                      | 8                   | 32                 | 256         | 1024       | 41%          |
| 39B                     | 64              | 8192        | 48            | 8                    | 2                      | 16                  | 32                 | 512         | 1536       | 41%          |
| 76B                     | 80              | 10240       | 60            | 8                    | 4                      | 32                  | 32                 | 1024        | 1792       | 43%          |
| 145B                    | 96              | 12288       | 80            | 8                    | 8                      | 64                  | 24                 | 1536        | 2304       | 44%          |
| 291B                    | 128             | 16384       | 90            | 8                    | 18                     | 144                 | 15                 | 2160        | 2430       | 45%          |
| 530B                    | 128             | 20480       | 105           | 8                    | 35                     | 280                 | 9                  | 2520        | 2520       | 49%          |
| 1T                      | 160             | 25600       | 128           | 8                    | 64                     | 512                 | 6                  | 3072        | 3072       | 49%          |



# Day 2 - Your turn !

## Outline of Day 2 ( 5.5 hours lunch break)

09:00 AM - 09:15 AM: Environment prep

09:15 AM - 09:45 AM: About compute

09:45 AM - 10:15 AM:

Lab 1 - Acquire data ( WebScraping )

Lab 1 - 2\_Estimate Time Needed for Compute

10:15 AM - 10:30 AM: Lab 1 - 3\_Megatron's core MPU (presentation)

**10:30 AM - 11:00 AM: Lab 1 - 3\_Megatron's core MPU (lab)**

11:00 AM - 11:40 AM:

Lab 1 - 4\_GPT Vocab+Merge file

Lab 1 - 5\_data\_preprocessing2mmap

11:40 AM - 12:00 PM: Intro to profiling

12:00 PM - 12:10 PM: Ask the Experts (Robert Dietrich)

12:10 PM - 13:10 PM: Lunch break

13:10 PM - 13:40 PM: Lab 1 - 6\_GPT training config vs GPUs performance

13:40 PM - 14:10 PM: Challenge overview

14:10 PM - 14:30 PM: Discussion & What's up next day

# MEGATRON MODEL PARALLISM

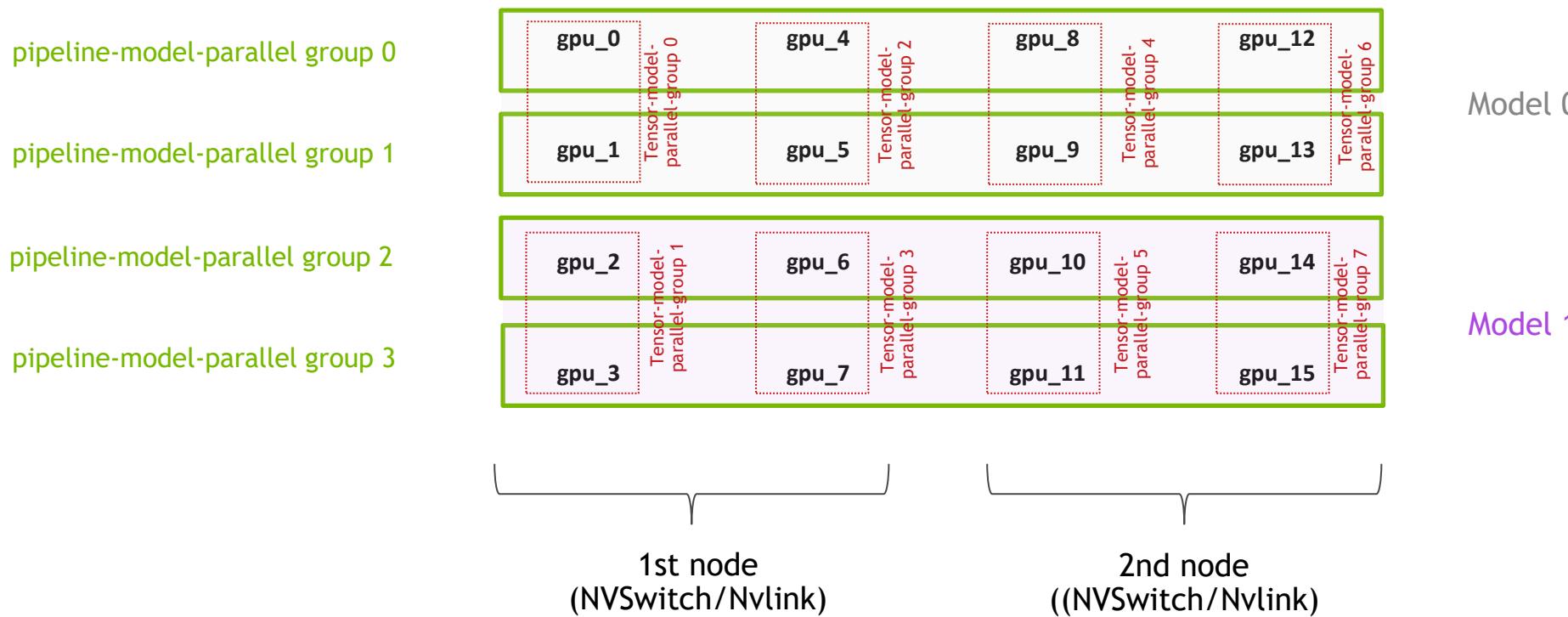
## GPU affinity grouping

Config set-up ---

Total number of GPUS : 16 GPUs ( 2 nodes , 8 gpus each node)

Pipeline-model-parallel : 4

Tensor-model-parallel : 2





# Day 2 - Run through Megatron GPT

## Outline of Day 2 ( 5.5 hours lunch break)

09:00 AM - 09:15 AM: Environment prep

09:15 AM - 09:45 AM: About compute

09:45 AM - 10:15 AM:

Lab 1 - Acquire data ( WebScraping )

Lab 1 - 2\_Estimate Time Needed for Compute

10:15 AM - 10:30 AM: Lab 1 - 3\_Megatron's core MPU (presentation)

10:30 AM - 11:00 AM: Lab 1 - 3\_Megatron's core MPU (lab)

11:00 AM - 11:40 AM:

Lab 1 - 4\_GPT Vocab+Merge file

Lab 1 - 5\_data\_preprocessing2mmap

11:40 AM - 12:00 PM: Intro to profiling

12:00 PM - 12:10 PM: Ask the Experts (Robert Dietrich)

12:10 PM - 13:10 PM: Lunch break

13:10 PM - 13:40 PM: Lab 1 - 6\_GPT training config vs GPUs performance

13:40 PM - 14:10 PM: Challenge overview

14:10 PM - 14:30 PM: Discussion & What's up next day

# PREPROCESSING DATA

- Input format: JSON array of dictionaries
  - One element per document
  - Text in the ‘text’ field by default (can also extract other fields)

## BERT

```
python tools/preprocess_data.py \  
    --input my-corpus.json \  
    --output-prefix my-bert \  
    --vocab bert-vocab.txt \  
    --dataset-impl mmap \  
    --tokenizer-type  
BertWordPieceLowerCase \  
    --split-sentences \  
    --workers 80
```

## GPT

```
python tools/preprocess_data.py \  
    --input my-corpus.json \  
    --output-prefix my-gpt \  
    --vocab gpt-vocab.txt \  
    --merge-file gpt-merges.txt \  
    --dataset-impl mmap \  
    --tokenizer-type GPT2BPETokenizer \  
    --append-eod \  
    --workers 80
```



# Day 2 - Your turn !

## Outline of Day 2 ( 5.5 hours lunch break)

09:00 AM - 09:15 AM: Environment prep

09:15 AM - 09:45 AM: About compute

09:45 AM - 10:15 AM:

Lab 1 - Acquire data ( WebScraping )

Lab 1 - 2\_Estimate Time Needed for Compute

10:15 AM - 10:30 AM: Lab 1 - 3\_Megatron's core MPU (presentation)

10:30 AM - 11:00 AM: Lab 1 - 3\_Megatron's core MPU (lab)

11:00 AM - 11:40 AM:

Lab 1 - 4\_GPT Vocab+Merge file

Lab 1 - 5\_data\_preprocessing2mmap

**11:40 AM - 12:00 PM: Intro to profiling**

12:00 PM - 12:10 PM: Ask the Experts (Robert Dietrich)

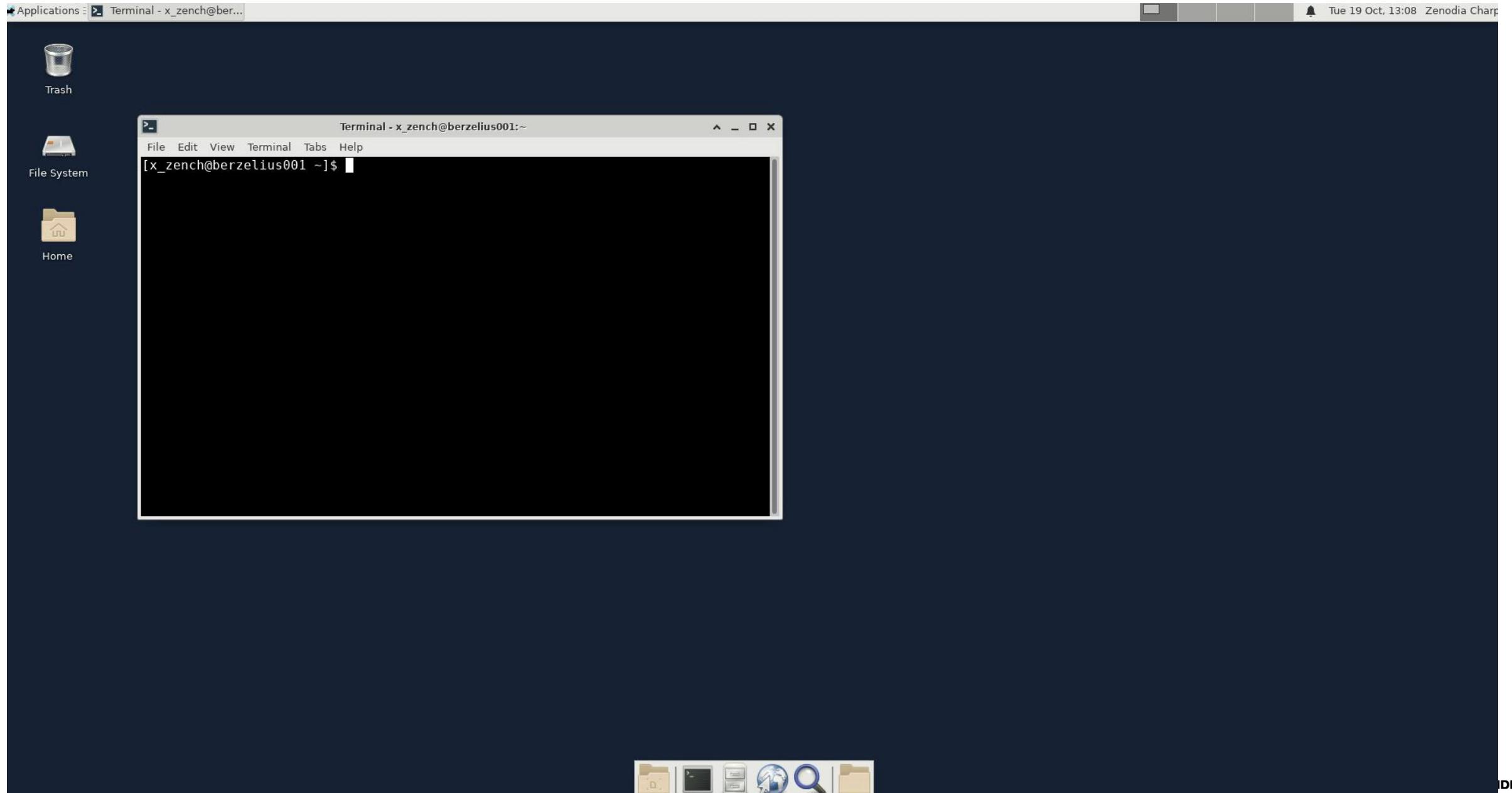
12:10 PM - 13:10 PM: Lunch break

13:10 PM - 13:40 PM: Lab 1 - 6\_GPT training config vs GPUs performance

13:40 PM - 14:10 PM: Challenge overview

14:10 PM - 14:30 PM: Discussion & What's up next day

# CALL OUT NSIGHT UI AND MONITOR ON BERZELIUS



Verify the code runs for a smaller GPT3 variants

Nsight Systems | Nsight Compute

NVTX for Tensorflow | NVTX for PyTorch | NVTX for MXNet

**F** PYTORCH mxnet

With NVTX tags on Nsight

Train [13.219 s]

Dense Block [834.622 μs]

Dense 1 [185.585 μs] Dense 2 [166.773...] Dense 3 [174.655 ...] Dense 4 [164.122...] Dense ...

NVTX  
[Default]  
Forward

MANY SHADES OF PROFILING  
(MEGATRON PROFILING DECK)



# Day 2 - Run through Megatron GPT

## Outline of Day 2 ( 5.5 hours lunch break)

09:00 AM - 09:15 AM: Environment prep

09:15 AM - 09:45 AM: About compute

09:45 AM - 10:15 AM:

Lab 1 - Acquire data ( WebScraping )

Lab 1 - 2\_Estimate Time Needed for Compute

10:15 AM - 10:30 AM: Lab 1 - 3\_Megatron's core MPU (presentation)

10:30 AM - 11:00 AM: Lab 1 - 3\_Megatron's core MPU (lab)

11:00 AM - 11:40 AM:

Lab 1 - 4\_GPT Vocab+Merge file

Lab 1 - 5\_data\_preprocessing2mmap

11:40 AM - 12:00 PM: Intro to profiling

**12:00 PM - 12:10 PM: Ask the Experts (Robert Dietrich)**

**12:10 PM - 13:10 PM: Lunch break**

13:10 PM - 13:40 PM: Lab 1 - 6\_GPT training config vs GPUs performance

13:40 PM - 14:10 PM: Challenge overview

14:10 PM - 14:30 PM: Discussion & What's up next day



# Day 2 - Your turn !

## Outline of Day 2 ( 5.5 hours lunch break)

09:00 AM - 09:15 AM: Environment prep

09:15 AM - 09:45 AM: About compute

09:45 AM - 10:15 AM:

Lab 1 - Acquire data ( WebScraping )

Lab 1 - 2\_Estimate Time Needed for Compute

10:15 AM - 10:30 AM: Lab 1 - 3\_Megatron's core MPU (presentation)

10:30 AM - 11:00 AM: Lab 1 - 3\_Megatron's core MPU (lab)

11:00 AM - 11:40 AM:

Lab 1 - 4\_GPT Vocab+Merge file

Lab 1 - 5\_data\_preprocessing2mmap

11:40 AM - 12:00 PM: Intro to profiling

12:00 PM - 12:10 PM: Ask the Experts (Robert Dietrich)

12:10 PM - 13:10 PM: Lunch break

**13:10 PM - 13:40 PM: Lab 1 - 6\_GPT training config vs GPUs performance**

**13:40 PM - 14:10 PM: Challenge overview**

14:10 PM - 14:30 PM: Discussion & What's up next day



# Day 2 - Run through Megatron GPT

## Outline of Day 2 ( 5.5 hours lunch break)

09:00 AM - 09:15 AM: Environment prep

09:15 AM - 09:45 AM: About compute

09:45 AM - 10:15 AM:

Lab 1 - Acquire data ( WebScraping )

Lab 1 - 2\_Estimate Time Needed for Compute

10:15 AM - 10:30 AM: Lab 1 - 3\_Megatron's core MPU (presentation)

10:30 AM - 11:00 AM: Lab 1 - 3\_Megatron's core MPU (lab)

11:00 AM - 11:40 AM:

Lab 1 - 4\_GPT Vocab+Merge file

Lab 1 - 5\_data\_preprocessing2mmap

11:40 AM - 12:00 PM: Intro to profiling

12:00 PM - 12:10 PM: Ask the Experts (Robert Dietrich)

12:10 PM - 13:10 PM: Lunch break

13:10 PM - 13:40 PM: Lab 1 - 6\_GPT training config vs GPUs performance

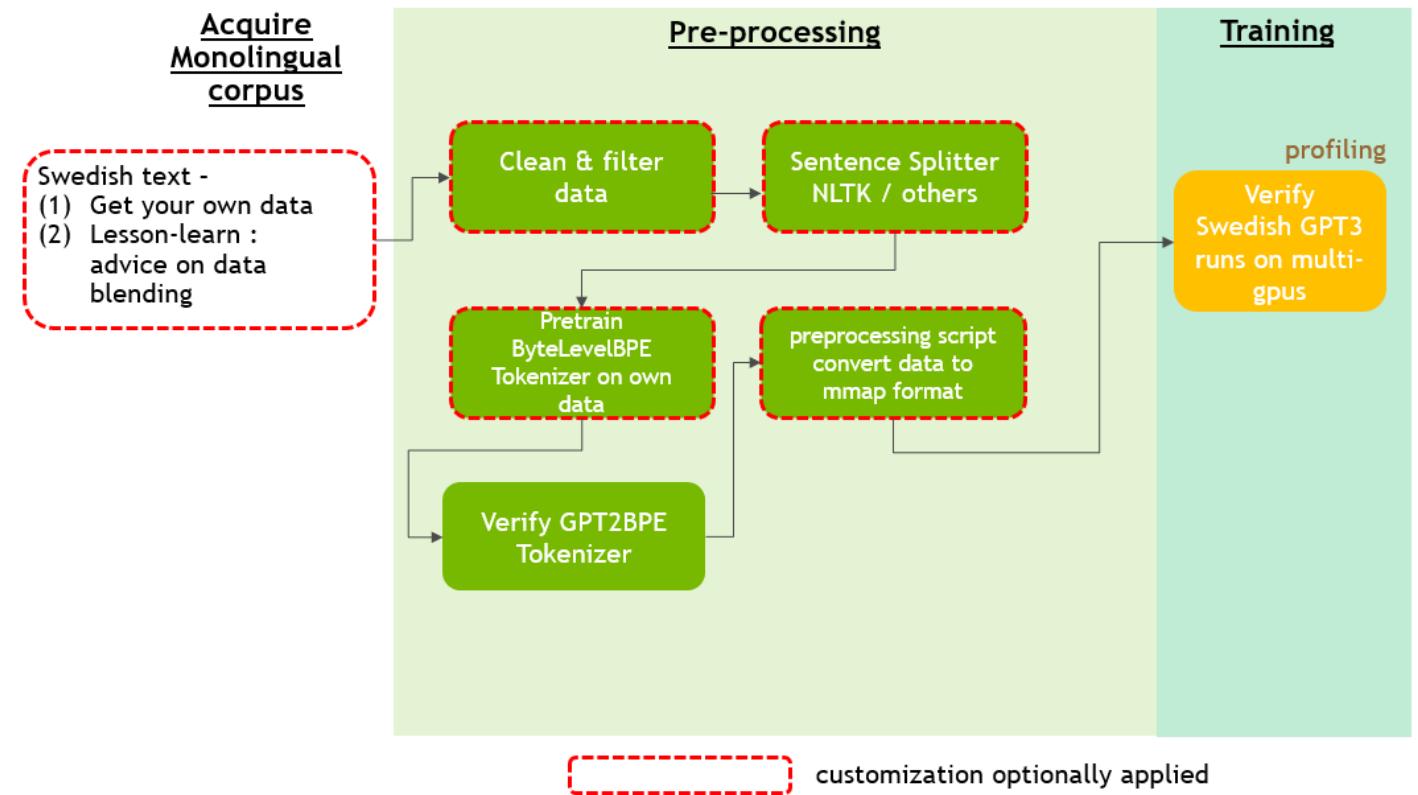
13:40 PM - 14:10 PM: Challenge overview

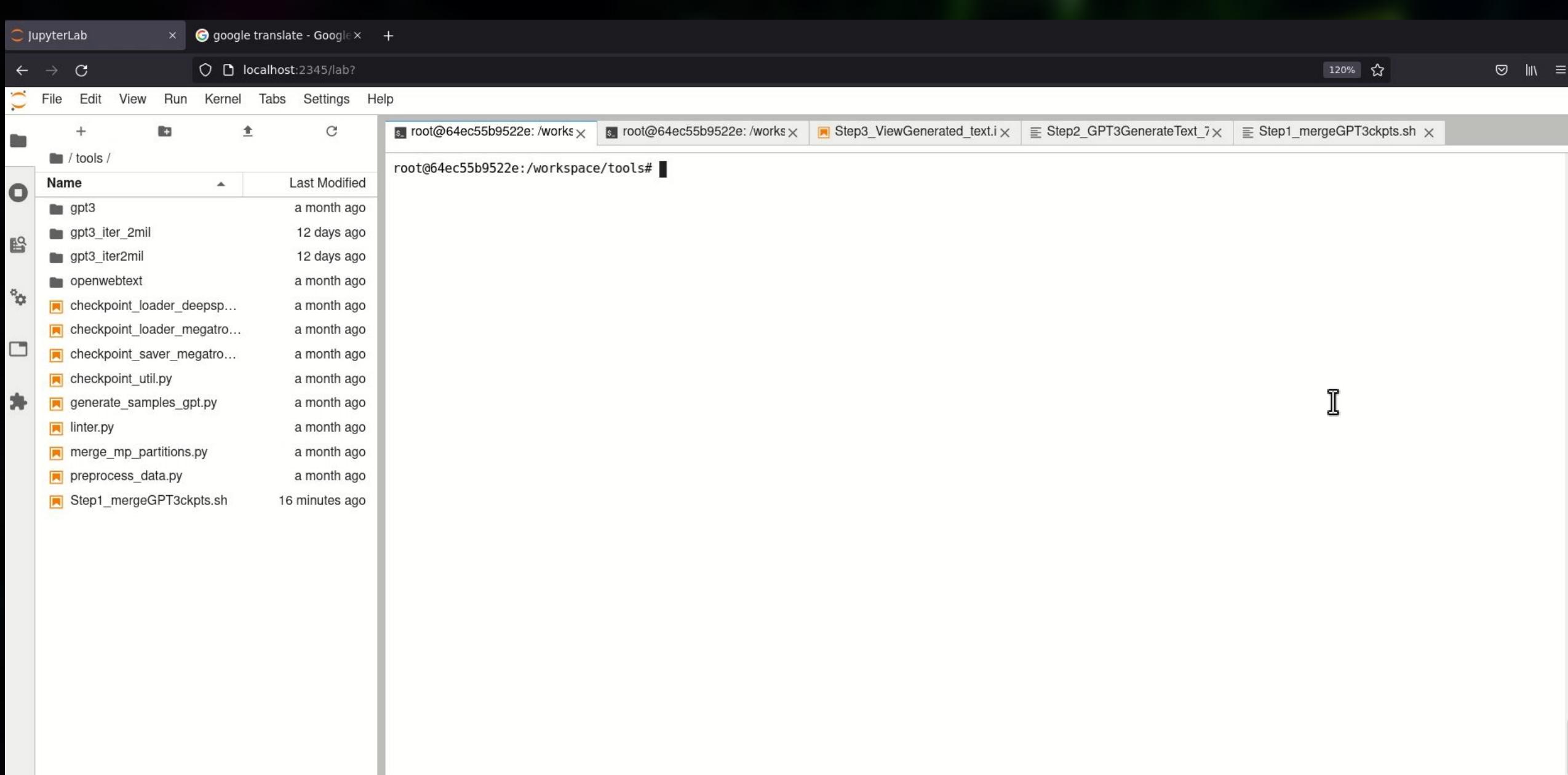
**14:10 PM - 14:30 PM: Discussion & What's up next day**

# End of Day 2 !



# Sneak peak of day 3 !





GENERATING TEXT WITH SWEDISH GPT 760M ITER\_1MIL

# INTRODUCING THE SPEAKERS OF THE DAY

Day 3's speakers - Denis Timonin &Avinash Kaur & Ashish Sardana



Dennis Timonin



Avinash Kaur



Ashish Sardana



# Day 3 - Customize Megatron for Swedish

## Outline of Day ( 6 hour with lunch break )

**09:00 AM - 09:15 AM: Recap and Overview of Day 3**

09:15 AM - 09:30 AM: About acquiring your own

09:30 AM - 10:20 AM:

Lab 2 - 1\_Acquiring data

Lab 2 - 2\_data cleaning & filter

10:20 AM - 10:30 AM: train own GPT tokenizer + preprocessing

10:30 AM - 11:00 AM: Lab 2 - 3\_train your own GPT tokenizer

11:00 AM - 11:30 AM: Lab 2 - 4\_data preprocessing

11:30 AM - 12:00 PM: Lab 2 - 4 Mini challenge

12:00 PM - 12:30 PM: Lunch Break

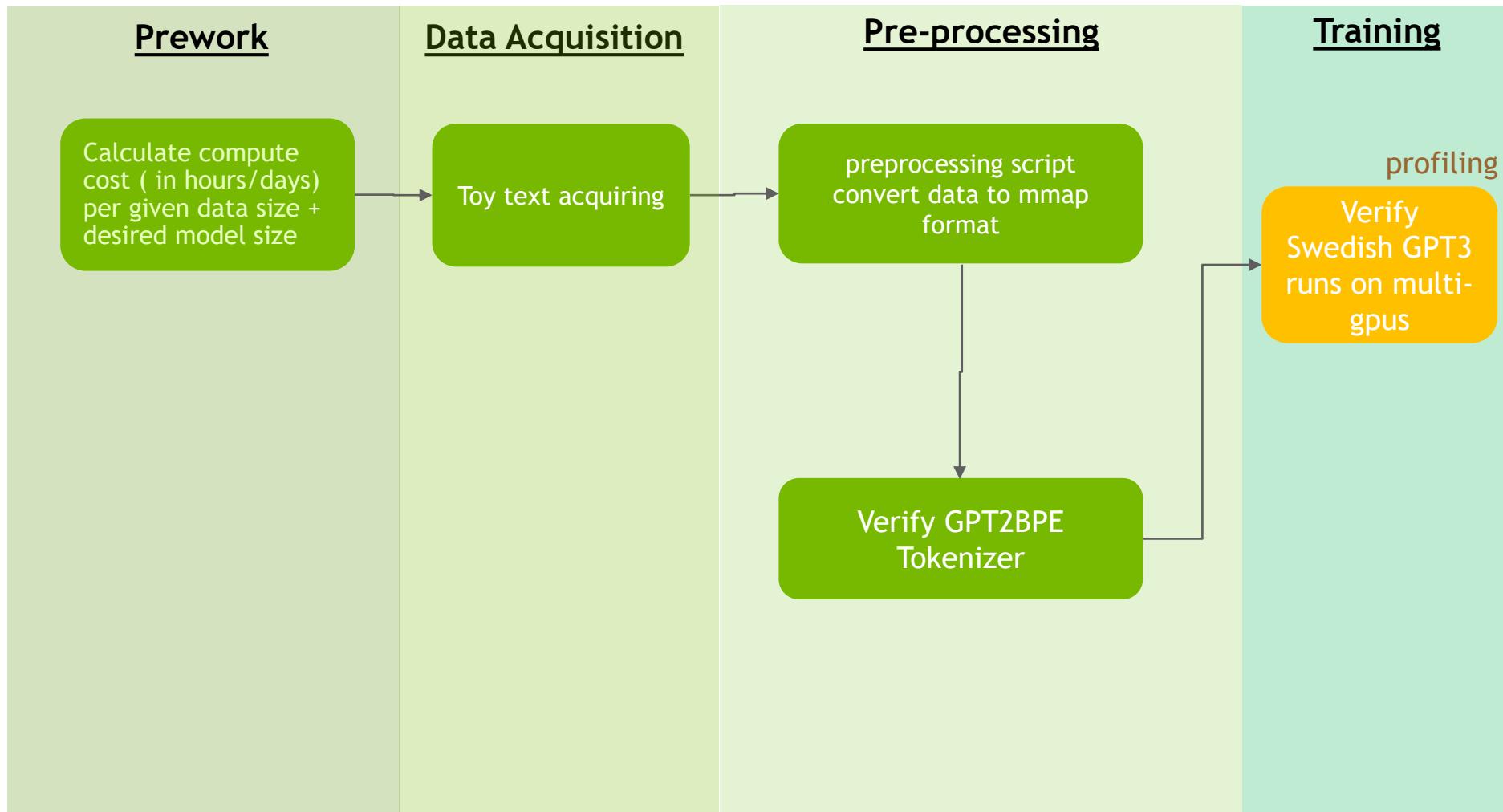
13:00 PM - 14:00 PM: Lab 2 - 5 Challenge - Go BIG or go home

14:00 PM - 14:30 PM: About deploying Big NLP model with NVIDIA Triton | faster transformer solution

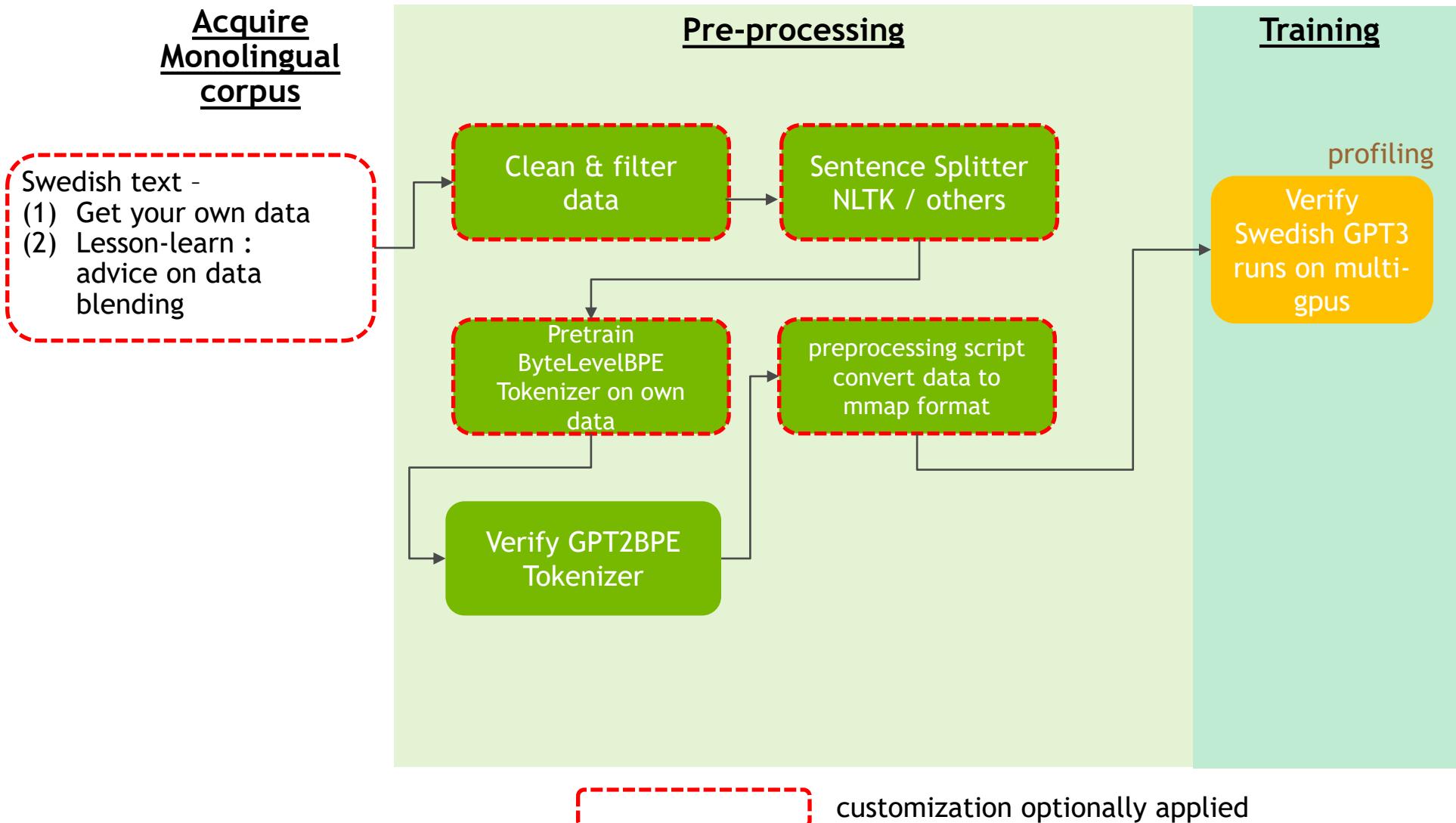
14:30 PM - 14:45 PM: Ask the Experts with NVIDIA data scientists on what's next for NLP

14:45 PM - 15:00 PM: Discussion & Final remarks

# DAY 2 - RECAP



# DAY 3 - OVERVIEW





# Day 3 - Customize Megatron for Swedish

## Outline of Day ( 6 hour with lunch break )

09:00 AM - 09:15 AM: Recap and Overview of Day 3

**09:15 AM - 09:30 AM: About acquiring your own + cleaning**

09:30 AM - 10:20 AM:

    Lab 2 - 1\_Acquiring data

    Lab 2 - 2\_data cleaning & filter

10:20 AM - 10:30 AM: train own GPT tokenizer + preprocessing

10:30 AM - 11:00 AM: Lab 2 - 3\_train your own GPT tokenizer

11:00 AM - 11:30 AM: Lab 2 - 4\_data preprocessing

11:30 AM - 12:00 PM: Lab 2 - 4 Mini challenge

12:00 PM - 12:30 PM: Lunch Break

13:00 PM - 14:00 PM: Lab 2 - 5 Challenge - Go BIG or go home

14:00 PM - 14:30 PM: About deploying Big NLP model with NVIDIA Triton | faster transformer solution

14:30 PM - 14:45 PM: Ask the Experts with NVIDIA data scientists on what's next for NLP

14:45 PM - 15:00 PM: Discussion & Final remarks

# ABOUT DATA ACQUISITION

Prioirtize getting large data volume then clean properly

Dataset characteristic -

- Make sure to carefully weight different datasets during model training, for example, commoncrawl data is known for having both high-quality and very bad quality data blended together. If one cannot clean out all the bad data, should consider sampling with less weight during training
- It is tempting to try on a very small dataset first, however, we advice to prioritize getting as much data as possible and do as much cleaning as possible, see section on advice on cleaning
- Get basic stats: # of tokens in the dataset | avg tokens per sentence | avg sentence count per doc

# SCRAP OWN PAGES TO OBTAIN OWN DATA

## Example of web pages scraping

Scrape webpages

```
!python link_extractor_js.py https://blogs.nvidia.com/blog/category/deep-learning/ -m 2
[*] Crawling: https://blogs.nvidia.com/blog/category/deep-learning/
[W:puppeteer.chromium_downloader] Starting Chromium download
100% [██████████] 109M/109M
[W:puppeteer.chromium_downloader] Chromium download done
[W:puppeteer.chromium_downloader] chromium extracted to /home/zcharpy/.cache/puppeteer/chromium/109M/109M
[*] Internal link: https://blogs.nvidia.com/blog/category/deep-learning/
[!] External link: https://www.nvidia.com/en-us/
[*] Internal link: https://blogs.nvidia.com/
[*] Internal link: https://blogs.nvidia.com/
[!] External link: https://twitter.com/nvidia
[!] External link: https://www.facebook.com/NVIDIA
[!] External link: https://news.ycombinator.com/submit
[!] External link: https://www.instagram.com/nvidia/
[!] External link: https://www.linkedin.com/company/nvidia/
[!] External link: https://www.reddit.com/r/nvidia/
[!] External link: https://feeds.feedburner.com/nvidia
[!] External link: https://www.twitch.tv/nvidia
[!] External link: https://www.youtube.com/nvidia
[*] Internal link: https://blogs.nvidia.com/privacy-c
[!] External link: https://www.nvidia.com/en-us/about
[!] External link: https://www.nvidia.com/en-us/contact
[*] Internal link: https://blogs.nvidia.com/blog/cate
[*] Internal link: https://blogs.nvidia.com/blog/tag/i
[*] Internal link: https://blogs.nvidia.com/blog/tag/i
[*] Internal link: https://blogs.nvidia.com/ai-podcast
[*] Internal link: https://blogs.nvidia.com/blog/2021
[*] Internal link: https://blogs.nvidia.com/blog/2021
graph/
[*] Internal link: https://blogs.nvidia.com/blog/2021/08/11/omniverse-making-of-gtc/
[*] Internal link: https://blogs.nvidia.com/blog/2021/08/11/lending-a-helping-hand-jules-anh-tuan-nguyen-on-building-a-neuroprosthetic/
[*] Internal link: https://blogs.nvidia.com/blog/2021/08/12/geforce-now-thursday-august-12/
[*] Internal link: https://blogs.nvidia.com/blog/2021/08/11/cloudxr-google-cloud-demo/
[*] Internal link: https://blogs.nvidia.com/blog/2021/08/10/omniverse-connector-extension-plugin-siggraph/
```

Parse text from individual webpages and concate into one .txt file

```
!head -1 /home/zcharpy/bootcamp/jupyter_notebook/Megatron-LM/dataset/EN/extractedNVblogs.txt
```

Scientists searching the universe for gravitational waves just got a boost thanks to a new study and AI. The research, recently published in *Nature Astronomy*, creates a deployable AI framework for detecting gravitational waves within massive amounts of data—at several magnitudes faster than real time. Created by a group of scientists from Argonne National Laboratory, the University of Chicago, the University of Illinois at Urbana-Champaign, NVIDIA, and IBM, the work highlights how AI and supercomputing can accelerate reproducible, data-driven discoveries.“As a computer scientist, what’s exciting to me about this project is that it shows how, with the right tools, AI methods can be integrated naturally into the workflows of scientists. Allowing them to do their work faster and better. Augmenting, not replacing, human intelligence,” study senior author Ian Foster, director of Argonne’s Data Science and Learning division said in a press release. In 2015, the advanced Laser Interferometer Gravitational-Wave Observatory (LIGO) first detected gravitational waves when two black holes, 1.3 billion light-years away, collided and merged. These waves occur when massive objects quickly accelerate (such as a star exploding or massive objects colliding) creating a ripple through space time. The notable discovery confirmed part of Einstein’s theory of relativity, hypothesizing that space and time are linked. It also marked the start of gravitational wave astronomy, which could result in a deeper understanding of the cosmos, including dark energy, gravity, and neutron stars. It also holds potential for scientists to step back through time to the moments around the Big Bang. Since 2015, many more gravitational wave sources have been detected from LIGO. As the observatory continues with sensor upgrades and refinements, the expanse of detectors within the universe will also grow, creating large amounts of data for processing. Quickly computing these data streams remains key to gravitational wave astronomy advancements and discoveries. In 2018 Elia Huerta, lead for Translational AI at Argonne, demonstrated the capability of machine learning to detect gravitational waves from multiple LIGO detector data streams. In this study, the researchers further refined the model, which uses the cuDNN-accelerated deep learning framework distributed over 64 NVIDIA GPUs. They tested the model against LIGO data from 2017 and found it accurately identified four binary black hole mergers—without any misclassifications. It also processed a month’s worth of data in under 7 minutes. “In this study, we’ve used the combined power of AI and supercomputing to help solve timely and relevant big-data experiments. We are now making AI studies fully reproducible, not merely ascertaining whether AI may provide a novel solution to grand challenges,” Huerta said. The team’s models are open-source and readily available. Read the full article in *Nature Astronomy* >>Read more >> Have a story to share? Submit an idea. Get the developer news feed straight to your inbox.

# CLEAN - REMOVE UNWANTED LANGUAGE

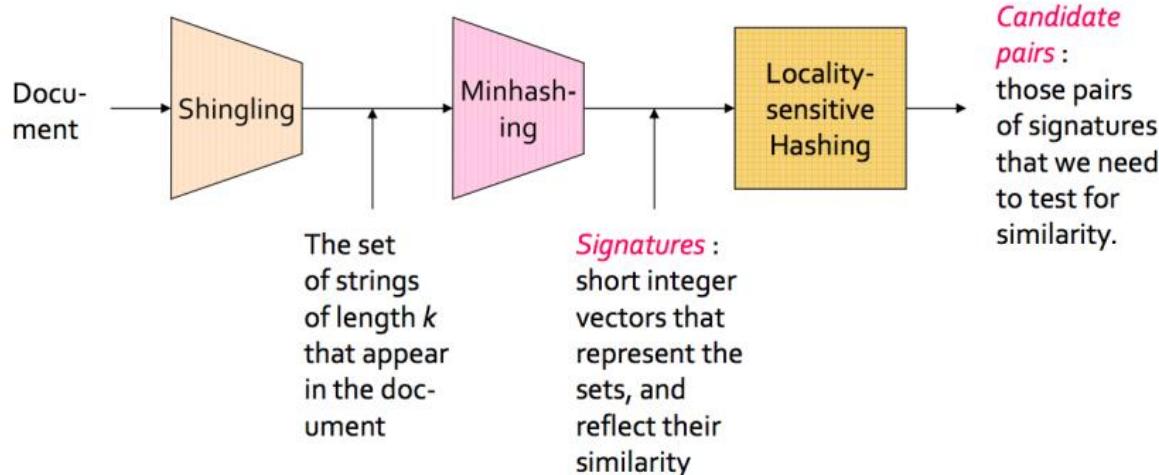
Remove unwanted language blended in the sentence of raw corpus

1. detect and filter undesired langauge in the raw text corpus

```
[1]: from langdetect import detect  
swe_raw_text='Under fredagsförmiddagen höll polis och räddningstjänst presskonferens tillsammans med en representanter från flygplatsens egna räddningsenhet och Örebro kommun.'  
detect(swe_raw_text)  
  
[1]: 'sv'  
  
[2]: danish_text='1. januar 2021 var folketallet 5.840.045. Ved den første folketælling i 1735 var der 718.000 danskere.'  
detect(danish_text)  
  
[2]: 'da'  
  
[3]: finnish_text='Jokaisella on oikeus vapaasti osallistua yhteiskunnan sivistyselämään, nauttia taiteista sekä päästä osalliseksi tieteen edistyksen mukanaan tuomista eduista.'  
detect(finnish_text)  
  
[3]: 'fi'
```

# CLEAN - METHODS OF DE-DUPLICATE

alternatives of computing similarity score

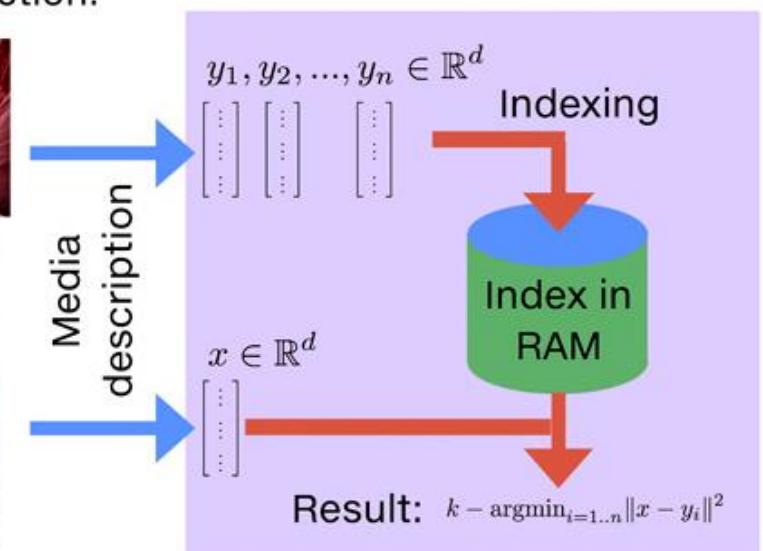


Source : [MinHash](#)

Build index for a collection:



Query:



Source : [Facebook Faiss](#)

# CLEAN - DEDUPLICATION

## Shingles → MinHash → LSH

deduplicate text based on similarity score

```
In [1]: import itertools
from lsh import cache, minhash # https://github.com/mattilyra/lsh

# a pure python shingling function that will be used in comparing
# LSH to true Jaccard similarities
def shingles(text, char_ngram=5):
    return set(text[head:head + char_ngram] for head in range(0, len(text) - char_ngram))

def jaccard(set_a, set_b):
    intersection = set_a & set_b
    union = set_a | set_b
    return len(intersection) / len(union)

def candidate_duplicates(document_feed, char_ngram=5, seeds=100, bands=5, hashbytes=4):
    char_ngram = char_ngram
    sims = []
    hasher = minhash.MinHasher(seeds=seeds, char_ngram=char_ngram, hashbytes=hashbytes)
    if seeds % bands != 0:
        raise ValueError('Seeds has to be a multiple of bands. {} % {} != 0'.format(seeds, bands))

    lshcache = cache.Cache(num_bands=bands, hasher=hasher)
    for i_line, line in enumerate(document_feed):
        line = line.decode('utf8')
        docid, headline_text = line.split('\t', 1)
        fingerprint = hasher.fingerprint(headline_text.encode('utf8'))

        # in addition to storing the fingerpring store the line
        # number and document ID to help analysis later on
        lshcache.add_fingerprint(fingerprint, doc_id=(i_line, docid))

    candidate_pairs = set()
    for b in lshcache.bins:
        for bucket_id in b:
            if len(b[bucket_id]) > 1:
                pairs_ = set(itertools.combinations(b[bucket_id], r=2))
                candidate_pairs.update(pairs_)

    return candidate_pairs
```

```
# reset file pointer and do LSH
fh.seek(0)
feed = itertools.islice(fh, 1000)

candidates = candidate_duplicates(feed, char_ngram=char_ngram, seeds=seeds, bands=bands, hashbytes=hashbytes)

# go over all the generated candidates comparing their similarities
similarities = []
for ((line_a, docid_a), (line_b, docid_b)) in candidates:
    doc_a, doc_b = lines[line_a], lines[line_b]
    shingles_a = shingles(lines[line_a])
    shingles_b = shingles(lines[line_b])

    jaccard_sim = jaccard(shingles_a, shingles_b)
    fingerprint_a = set(hasher.fingerprint(doc_a.encode('utf8')))
    fingerprint_b = set(hasher.fingerprint(doc_b.encode('utf8')))
    minhash_sim = len(fingerprint_a & fingerprint_b) / len(fingerprint_a | fingerprint_b)
    similarities.append((docid_a, docid_b, jaccard_sim, minhash_sim))

for a,b,jsim,msim in random.sample(similarities, k=2):
    print("pair of similar sentences with jaccard_sim score:{} and minhash_sim score:{} --- \n".format(str(jsim),str(msim)))
    a=int(a)
    b=int(b)
    text_a=df2.iloc[a,1]
    text_b=df2.iloc[b,2]
    if text_a==text_b:
        print("100% duplicates \n")
        print("text_a:", text_a.split(' ')[:5])
        print("text_b:", text_b.split(' ')[:5])
        print('-----*10')
        import random

print('\nThere are **{}** candidate duplicates in total\n'.format(len(candidates)))
random.sample(similarities, k=1)

pair of similar sentences with jaccard_sim score:0.6978398983481575 and minhash_sim score:0.5037593984962406 ---
text_a: ['Researchers,', 'developers,', 'and', 'engineers', 'worldwide']
text_b: ['Edge', 'computing', 'has', 'been', 'around']
-----
pair of similar sentences with jaccard_sim score:0.9133693568066934 and minhash_sim score:0.9047619047619048 ---
text_a: ['The', 'NGC', 'team', 'is', 'hosting']
text_b: ['The', 'first', 'post', 'in', 'this']
-----

There are **31** candidate duplicates in total
```

# CLEAN - SENTENCE BOUNDARY

## alternatives of sentence-splitter for Swedish

```
[1]: import nltk
from nltk.tokenize import sent_tokenize
text='Har någon funderat på varför man inte får inomhustemperaturens kurva synlig i grafen? Är det någon som frågat Thermia? Skulle det inte vara väsentligt att kunna kolla historiken på den då man skall ställa in kurvan?'
print("original doc is :\n ", text)
sents=sent_tokenize(text)
i=0
for sent in sents:
    print("----- sentence {} -----".format(str(i)))
    print(sent)
    i+=1

original doc is :
 Har någon funderat på varför man inte får inomhustemperaturens kurva synlig i grafen? Är det någon som frågat Thermia? Skulle det inte vara väsentligt att kunna kolla historiken på den då man skall ställa in kurvan?
----- sentence 0 -----
Har någon funderat på varför man inte får inomhustemperaturens kurva synlig i grafen?
----- sentence 1 -----
Är det någon som frågat Thermia?
----- sentence 2 -----
Skulle det inte vara väsentligt att kunna kolla historiken på den då man skall ställa in kurvan?

[2]: from sentence_splitter import SentenceSplitter, split_text_into_sentences
splitter = SentenceSplitter(language='sv', non_breaking_prefix_file='custom_eng')
sents=splitter.split(text=text)
i=0
for sent in sents:
    print("----- sentence {} -----".format(str(i)))
    print(sent)
    i+=1

----- sentence 0 -----
Har någon funderat på varför man inte får inomhustemperaturens kurva synlig i grafen?
----- sentence 1 -----
Är det någon som frågat Thermia?
----- sentence 2 -----
Skulle det inte vara väsentligt att kunna kolla historiken på den då man skall ställa in kurvan?

[45]: import re
def custom_sentence_cutter(text):
    sents=re.split('|\.\.',text)
    return sents
sents=custom_sentence_cutter(sv[rn])
sents=[s for s in sents if len(s)>1]
i=0
for sent in sents:
    print("----- sentence {} -----".format(str(i)))
    print(sent)
    i+=1

----- sentence 0 -----
Hela anslutningsförfarandet har präglats av andra krav som vi menar är oacceptabla
----- sentence 1 -----
t
----- sentence 2 -----
ex
----- sentence 3 -----
diskriminering när det gäller rörligheten för arbetstagare och lika tillgång till EU-finansiering liksom de ensidiga skyddsclauserna som kan användas mot dessa länders
----- sentence 4 -----
och enbart dessa ländernas
----- sentence 5 -----
intressen
```

# [DEMO] DATA CLEANING AND FILTERING

The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** File Edit View Run Kernel Tabs Settings Help
- Tab Bar:** root@dd533a7e3b90:/works x data\_cleaning\_demo.ipynb x Python 3 C
- Toolbar:** + X □ ▶ ■ C ▶ Code ▾
- Section Header:** get the blended dataset
- Code Cell 1:** Python code to read text files from a directory, identify them by language, and print their lengths.

```
[ ]: import random
import os
txt_dir='./txt/'
text_files=os.listdir(txt_dir)
text_files=[t for t in text_files if t.endswith('.txt')]
tot=0
def get_lines(f_name):
    f=open(txt_dir+f_name,'r')
    lines=f.readlines()
    return lines

for cur_f in text_files:
    if cur_f.endswith('.txt'):
        if 'da' in cur_f:
            da_line=get_lines(cur_f)
            print("Dannish ", len(da_line))
        elif 'fi' in cur_f :
            fi_line=get_lines(cur_f)
            print("Finnish ", len(fi_line))
        elif 'de' in cur_f:
            de_line=get_lines(cur_f)
            print('German ', len(de_line))
        else:
            sv_line=get_lines(cur_f)
            print("Swedish ", len(sv_line))
```

- Code Cell 2:** Python code to sample lines from specific languages for filtering.

```
[ ]: sv=random.sample(sv_line,100)
da=random.sample(da_line,573) # these are the language we want to filtered out
de=random.sample(de_line,611) # these are the language we want to filtered out
fi=random.sample(fi_line,520) # these are the language we want to filtered out
```

# DATA COLLECTION

## Dataset Weighting

| Dataset           | Tokens (billions) | Weights (%) | Epochs |
|-------------------|-------------------|-------------|--------|
| Books3            | 25.7              | 14.3        | 1.5    |
| OpenWebText2      | 14.8              | 19.3        | 3.6    |
| Stack Exchange    | 11.6              | 5.7         | 1.4    |
| PubMed Abstracts  | 4.4               | 2.9         | 1.8    |
| Wikipedia         | 4.2               | 4.8         | 3.2    |
| Gutenberg (PG-19) | 2.7               | 0.9         | 0.9    |
| BookCorpus2       | 1.5               | 1.0         | 1.8    |
| NIH ExPorter      | 0.3               | 0.2         | 1.8    |
| Pile-CC           | 49.8              | 9.4         | 0.5    |
| ArXiv             | 20.8              | 1.4         | 0.2    |
| GitHub            | 24.3              | 1.6         | 0.2    |
| CC-2020-50        | 68.7              | 13.0        | 0.5    |
| CC-2021-04        | 82.6              | 15.7        | 0.5    |
| RealNews          | 21.9              | 9.0         | 1.1    |
| CC-Stories        | 5.3               | 0.9         | 0.5    |

# DATA CLEANSING ADVISE

## Summary

1. Prioritize getting as large raw dataset as possible
2. Clean it aggressively - OpenAI raw data is **45TB** --> after cleaned is **570GB**
  1. Detect and remove undesired language inside the raw corpus
  2. Do NOT include downstream tasks' datasets into training dataset, deduplicate training dataset against downstream datasets
  3. Find best suited alternatives to de-duplication at the **document** level, within and across datasets, to prevent redundancy and preserve the integrity of original dataset's characteristics
  4. Add known high-quality reference corpora to the training mix whenever possible
  5. Consider filter by # of characters in a sentences within a document
3. Megatron-LM repo provide some basic [data cleansing scripts](#)

source: <https://arxiv.org/pdf/2005.14165.pdf>



# Day 3 - Your turn !

## Outline of Day ( 6 hour with lunch break )

09:00 AM - 09:15 AM: Recap and Overview of Day 3

09:15 AM - 09:30 AM: About acquiring your own

**09:30 AM - 10:20 AM:**

**Lab 2 - 1\_Acquiring data**

**Lab 2 - 2\_data cleaning & filter**

10:20 AM - 10:30 AM: train own GPT tokenizer + data preprocessing

10:30 AM - 11:00 AM: Lab 2 - 3\_train your own GPT tokenizer

11:00 AM - 11:30 AM: Lab 2 - 4\_data preprocessing

11:30 AM - 12:00 PM: Lab 2 - 4 Mini challenge

12:00 PM - 12:30 PM: Lunch Break

13:00 PM - 14:00 PM: Lab 2 - 5 Challenge - Go BIG or go home

14:00 PM - 14:30 PM: About deploying Big NLP model with NVIDIA Triton | faster transformer solution

14:30 PM - 14:45 PM: Ask the Experts with NVIDIA data scientists on what's next for NLP

14:45 PM - 15:00 PM: Discussion & Final remarks



# Day 3 - Customize Megatron for Swedish

## Outline of Day ( 6 hour with lunch break )

09:00 AM - 09:15 AM: Recap and Overview of Day 3

09:15 AM - 09:30 AM: About acquiring your own

09:30 AM - 10:20 AM:

    Lab 2 - 1\_Acquiring data

    Lab 2 - 2\_data cleaning & filter

**10:20 AM - 10:30 AM: train own GPT tokenizer + data preprocessing**

10:30 AM - 11:00 AM: Lab 2 - 3\_train your own GPT tokenizer

11:00 AM - 11:30 AM: Lab 2 - 4\_data preprocessing

11:30 AM - 12:00 PM: Lab 2 - 4 Mini challenge

12:00 PM - 12:30 PM: Lunch Break

13:00 PM - 14:00 PM: Lab 2 - 5 Challenge - Go BIG or go home

14:00 PM - 14:30 PM: About deploying Big NLP model with NVIDIA Triton | faster transformer solution

14:30 PM - 14:45 PM: Ask the Experts with NVIDIA data scientists on what's next for NLP

14:45 PM - 15:00 PM: Discussion & Final remarks

# TRAIN YOUR OWN GPT3 VOCAB + MERGE

## train your own GPT compatible tokenizer

```
import os , sys
from tokenizers import Tokenizer
from tokenizers import Tokenizer, models, pre_tokenizers, trainers
from tokenizers.decoders import ByteLevel as ByteLevelDecoder ←
from tokenizers.trainers import BpeTrainer
from tokenizers.normalizers import NFKC, Sequence
from tokenizers.pre_tokenizers import Whitespace
import argparse
import os

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--infile', default=None, type=str, help='path to the txt files')
    parser.add_argument('--bpe_path', default=None, type=str, help='output bpe path')
    parser.add_argument('--load_pretrained', action='store_true', help='load pretrained BPE model')
    parser.add_argument('--incl_special_toks', action='store_true', help='load pretrained BPE model')
    parser.add_argument('--vocab_size', default=None, type=int,
                        help='specify the vocab_size when training HF BPE for chinese usually 8k/16k/32k/48k/64k')
    args = parser.parse_args()
    tokenizer = Tokenizer(models.BPE())
    if args.load_pretrained :
        print("loading gpt2bpe english vocab and merge \n")
        vocab_file='/workspace/SVdata/gpt2bpe/gpt2-vocab.json'
        merge_file='/workspace/SVdata/gpt2bpe/gpt2-merges.txt'
        tokenizer.model = models.BPE.from_file(vocab_file, merge_file)

    tokenizer.pre_tokenizer = pre_tokenizers.ByteLevel() ←
    tokenizer.decoder = ByteLevelDecoder() ←
    if args.incl_special_toks:
        special_tokens = ['[CLS]', '[BOS]', '[EOS]', '[SEP]', '[PAD]', '[MASK]', '[UNK]', '[EOD]']
        print("including special tokens : ", special_tokens)
    else:
        print("include minimal special token end of text ")
        special_tokens= ["<|endoftext|>"] ←

    # Set the training hyperparameters
    trainer = trainers.BpeTrainer(
        vocab_size=args.vocab_size,
        special_tokens=special_tokens,
        initial_alphabet=pre_tokenizers.ByteLevel.alphabet() ←
    )
    # Train it with either files or an iterator:
    tokenizer.train([args.infile], trainer=trainer)
    print("Trained vocab size: {}".format(tokenizer.get_vocab_size()))
    # You will see the generated files in the output.
    print("saving trained BPE model to : ", args.bpe_path)
    tokenizer.model.save(args.bpe_path)
    print("model saved ! \n\n")
```

# PREPROCESSING DATA

- Input format: JSON array of dictionaries
  - One element per document
  - Text in the ‘text’ field by default (can also extract other fields)

## BERT

```
python tools/preprocess_data.py \  
    --input my-corpus.json \  
    --output-prefix my-bert \  
    --vocab bert-vocab.txt \  
    --dataset-impl mmap \  
    --tokenizer-type BertWordPieceLowerCase \  
    --split-sentences \  
    --workers 80
```

## GPT

```
python tools/preprocess_data.py \  
    --input my-corpus.json \  
    --output-prefix my-gpt \  
    --vocab gpt-vocab.txt \  
    --merge-file gpt-merges.txt \  
    --dataset-impl mmap \  
    --tokenizer-type GPT2BPETokenizer \  
    --append-eod \  
    --workers 80
```



# Day 3 - Your turn !

## Outline of Day ( 6 hour with lunch break )

09:00 AM - 09:15 AM: Recap and Overview of Day 3

09:15 AM - 09:30 AM: About acquiring your own

09:30 AM - 10:20 AM:

    Lab 2 - 1\_Acquiring data

    Lab 2 - 2\_data cleaning & filter

10:20 AM - 10:30 AM: train own GPT tokenizer + data preprocessing

**10:30 AM - 11:00 AM: Lab 2 - 3\_train your own GPT tokenizer**

**11:00 AM - 11:30 AM: Lab 2 - 4\_data preprocessing**

**11:30 AM - 12:00 PM: Lab 2 - 4 Mini challenge**

12:00 PM - 12:30 PM: Lunch Break

13:00 PM - 14:00 PM: Lab 2 - 5 Challenge - Go BIG or go home

14:00 PM - 14:30 PM: About deploying Big NLP model with NVIDIA Triton | faster transformer solution

14:30 PM - 14:45 PM: Ask the Experts with NVIDIA data scientists on what's next for NLP

14:45 PM - 15:00 PM: Discussion & Final remarks



# Day 3 - Customize Megatron for Swedish

## Outline of Day ( 6 hour with lunch break )

09:00 AM - 09:15 AM: Recap and Overview of Day 3

09:15 AM - 09:30 AM: About acquiring your own

09:30 AM - 10:20 AM:

    Lab 2 - 1\_Acquiring data

    Lab 2 - 2\_data cleaning & filter

10:20 AM - 10:30 AM: train own GPT tokenizer + data preprocessing

10:30 AM - 11:00 AM: Lab 2 - 3\_train your own GPT tokenizer

11:00 AM - 11:30 AM: Lab 2 - 4\_data preprocessing

11:30 AM - 12:00 PM: Lab 2 - 4 Mini challenge

**12:00 PM - 12:30 PM: Lunch Break**

13:00 PM - 14:00 PM: Lab 2 - 5 Challenge - Go BIG or go home

14:00 PM - 14:30 PM: About deploying Big NLP model with NVIDIA Triton | faster transformer solution

14:30 PM - 14:45 PM: Ask the Experts with NVIDIA data scientists on what's next for NLP

14:45 PM - 15:00 PM: Discussion & Final remarks



# Day 3 - Customize Megatron for Swedish

## Outline of Day ( 6 hour with lunch break )

09:00 AM - 09:15 AM: Recap and Overview of Day 3

09:15 AM - 09:30 AM: About acquiring your own

09:30 AM - 10:20 AM:

    Lab 2 - 1\_Acquiring data

    Lab 2 - 2\_data cleaning & filter

10:20 AM - 10:30 AM: train own GPT tokenizer + data preprocessing

10:30 AM - 11:00 AM: Lab 2 - 3\_train your own GPT tokenizer

11:00 AM - 11:30 AM: Lab 2 - 4\_data preprocessing

11:30 AM - 12:00 PM: Lab 2 - 4 Mini challenge

12:00 PM - 12:30 PM: Lunch Break

**13:00 PM - 14:00 PM: Lab 2 - 5 Challenge - Go BIG or go home (hint)**

14:00 PM - 14:30 PM: About deploying Big NLP model with NVIDIA Triton | faster transformer solution

14:30 PM - 14:45 PM: Ask the Experts with NVIDIA data scientists on what's next for NLP

14:45 PM - 15:00 PM: Discussion & Final remarks

# MEGATRON COMMAND LINE ARGUMENTS

## Model Architecture Arguments

|                           |                              |
|---------------------------|------------------------------|
| --num-layers              | Number of transformer layers |
| --hidden-size             | Hidden size between layers   |
| --num-attention-heads     | Number of attention heads    |
| --seq-length              | Maximum sequence length      |
| --max-position-embeddings | Maximum position embeddings  |

## Distributed training arguments

|   |                                |
|---|--------------------------------|
| --tensor-model-parallel-size            | Degree of tensor parallelism   |
| --pipeline-model-parallel-size          | Degree of pipeline parallelism |
| --num-layers-per-virtual-pipeline-stage | Allows interleaved schedule    |

# MEGATRON COMMAND LINE ARGUMENTS

## Training Arguments

|                     |  |
|---------------------|--|
| --micro-batch-size  | Batch size passed through the model  |
| --global-batch-size | Batch size of each iteration, including data parallelism and gradient accumulation |
| --fp16              | Train with fp16, required to fit any reasonably large model                        |

# MEGATRON COMMAND LINE ARGUMENTS

## dataset Arguments

either

**--train-iters**

or

**--train-samples**



# Day 3 - Your Turn !

## Outline of Day ( 6 hour with lunch break )

09:00 AM - 09:15 AM: Recap and Overview of Day 3

09:15 AM - 09:30 AM: About acquiring your own

09:30 AM - 10:20 AM:

    Lab 2 - 1\_Acquiring data

    Lab 2 - 2\_data cleaning & filter

10:20 AM - 10:30 AM: train own GPT tokenizer + data preprocessing

10:30 AM - 11:00 AM: Lab 2 - 3\_train your own GPT tokenizer

11:00 AM - 11:30 AM: Lab 2 - 4\_data preprocessing

11:30 AM - 12:00 PM: Lab 2 - 4 Mini challenge

12:00 PM - 13:00 PM: Lunch Break

**13:00 PM - 14:00 PM: Lab 2 - 5 Challenge - Go BIG or go home (lab)**

14:00 PM - 14:30 PM: About deploying Big NLP model with NVIDIA Triton | faster transformer solution

14:30 PM - 14:45 PM: Ask the Experts with NVIDIA data scientists on what's next for NLP

14:45 PM - 15:00 PM: Discussion & Final remarks



# Day 3 - Customize Megatron for Swedish

## Outline of Day ( 6 hour with lunch break )

09:00 AM - 09:15 AM: Recap and Overview of Day 3

09:15 AM - 09:30 AM: About acquiring your own

09:30 AM - 10:20 AM:

Lab 2 - 1\_Acquiring data

Lab 2 - 2\_data cleaning & filter

10:20 AM - 10:30 AM: train own GPT tokenizer + data preprocessing

10:30 AM - 11:00 AM: Lab 2 - 3\_train your own GPT tokenizer

11:00 AM - 11:30 AM: Lab 2 - 4\_data preprocessing

11:30 AM - 12:00 PM: Lab 2 - 4 Mini challenge

12:00 PM - 12:30 PM: Lunch Break

13:00 PM - 13:45 PM: Lab 2 - 5 Challenge - Go BIG or go home (lab)

13:45 PM - 14:15 PM: About deploying Big NLP model with NVIDIA Triton | faster transformer solution

14:15 PM - 14:45 PM: Ask the Experts with NVIDIA data scientists on what's next for NLP

**14:45 PM - 15:00 PM: Discussion & Final remarks**



# Day 3 - Customize Megatron for Swedish

## Outline of Day ( 6 hour with lunch break )

09:00 AM - 09:15 AM: Recap and Overview of Day 3

09:15 AM - 09:30 AM: About acquiring your own

09:30 AM - 10:20 AM:

Lab 2 - 1\_Acquiring data

Lab 2 - 2\_data cleaning & filter

10:20 AM - 10:30 AM: train own GPT tokenizer + data preprocessing

10:30 AM - 11:00 AM: Lab 2 - 3\_train your own GPT tokenizer

11:00 AM - 11:30 AM: Lab 2 - 4\_data preprocessing

11:30 AM - 12:00 PM: Lab 2 - 4 Mini challenge

12:00 PM - 12:30 PM: Lunch Break

13:00 PM - 13:30 PM: Lab 2 - 5 Challenge - Go BIG or go home (lab)

13:30 PM - 14:00 PM: About deploying Big NLP model with NVIDIA Triton | faster transformer solution

14:15 PM - 14:45 PM: Ask the Experts with NVIDIA data scientists on what's next for NLP

**14:45 PM - 15:00 PM: Discussion & Final remarks**



# Day 3 - Customize Megatron for Swedish

## Outline of Day ( 6 hour with lunch break )

09:00 AM - 09:15 AM: Recap and Overview of Day 3

09:15 AM - 09:30 AM: About acquiring your own

09:30 AM - 10:20 AM:

Lab 2 - 1\_Acquiring data

Lab 2 - 2\_data cleaning & filter

10:20 AM - 10:30 AM: train own GPT tokenizer + data preprocessing

10:30 AM - 11:00 AM: Lab 2 - 3\_train your own GPT tokenizer

11:00 AM - 11:30 AM: Lab 2 - 4\_data preprocessing

11:30 AM - 12:00 PM: Lab 2 - 4 Mini challenge

12:00 PM - 12:30 PM: Lunch Break

13:00 PM - 13:45 PM: Lab 2 - 5 Challenge - Go BIG or go home (lab)

13:45 PM - 14:15 PM: About deploying Big NLP model with NVIDIA Triton | faster transformer solution

14:15 PM - 14:45 PM: Ask the Experts with NVIDIA data scientists on what's next for NLP

**14:45 PM - 15:00 PM: Discussion & Final remarks**

# MEGATRON 760M GPT TEXT GENERATION

## Training Arguments

The screenshot shows a Jupyter Notebook interface with several tabs at the top: 'root@64ec55b9522e: /works <' (grey), 'view\_generated\_text.ipynb <' (orange), 'GPT3GenerateText\_760Mgp <' (grey), '5a\_GeneratingGPTtext\_step <' (grey), and 'generate\_samples\_gpt.py <' (grey). The main area is a code cell [1] containing Python code to read JSON samples and print generated text. The output shows three samples:

```
[1]: import json
f=open('samples_760Miter1mil.json','r')
ls=[json.loads(l) for l in f.readlines()]
i=0
for txt in [item['text'] for item in ls]:
    print("generated text sample {} : {}".format(str(i),txt))
    i+=1
```

generated text sample 0 : Produkter med detta motiv Vit I Love Can - over stone T-

generated text sample 1 : Raspberry Ketone Plus verkar vara potenta vikt förlust och vikt hantering komplement som kan motivera en massa män i Kristianstad Sverige anta en hälsosammare mer aktiv livsstil och därmed få alla fördelar denna hallon keton extrahera så rikligt ger

generated text sample 2 : Restauranger i närheten av Iforway

generated text sample 3 : Sök efter hotell i

# JOINING ENCCS HACKATHON

ENCCS hackathon Dec 6th 2021 ( application deadline Oct 31st )



<https://www.gphackathons.org/event/enccs-gpu-hackathon-2021>



# BRILLIANT MINDS. BREAKTHROUGH DISCOVERIES.

The Conference for AI Innovators, Technologists,  
and Creators

Register now at [www.nvidia.com/GTC](http://www.nvidia.com/GTC)

**A Zero-code Approach to Creating Production-ready AI Models [A31176]**

**NVIDIA NeMo: Speech Recognition, Speech Synthesis, and NLP Updates [A31556]**

**Conversational AI Demystified, and a Hands-on Walkthrough [A31081]**

**How NVIDIA IT Uses NVIDIA RIVA for Conversational AI Services [A31510]**

**A Step-by-step Guide to Building Large Custom Language Models [A31082]**

**Technical Leaders in Natural Language Processing [A31080]**

NOVEMBER 8-11, 2021 | [www.nvidia.com/GTC](http://www.nvidia.com/GTC)

