LUMI: BERT in an Hour, GPT in a Week

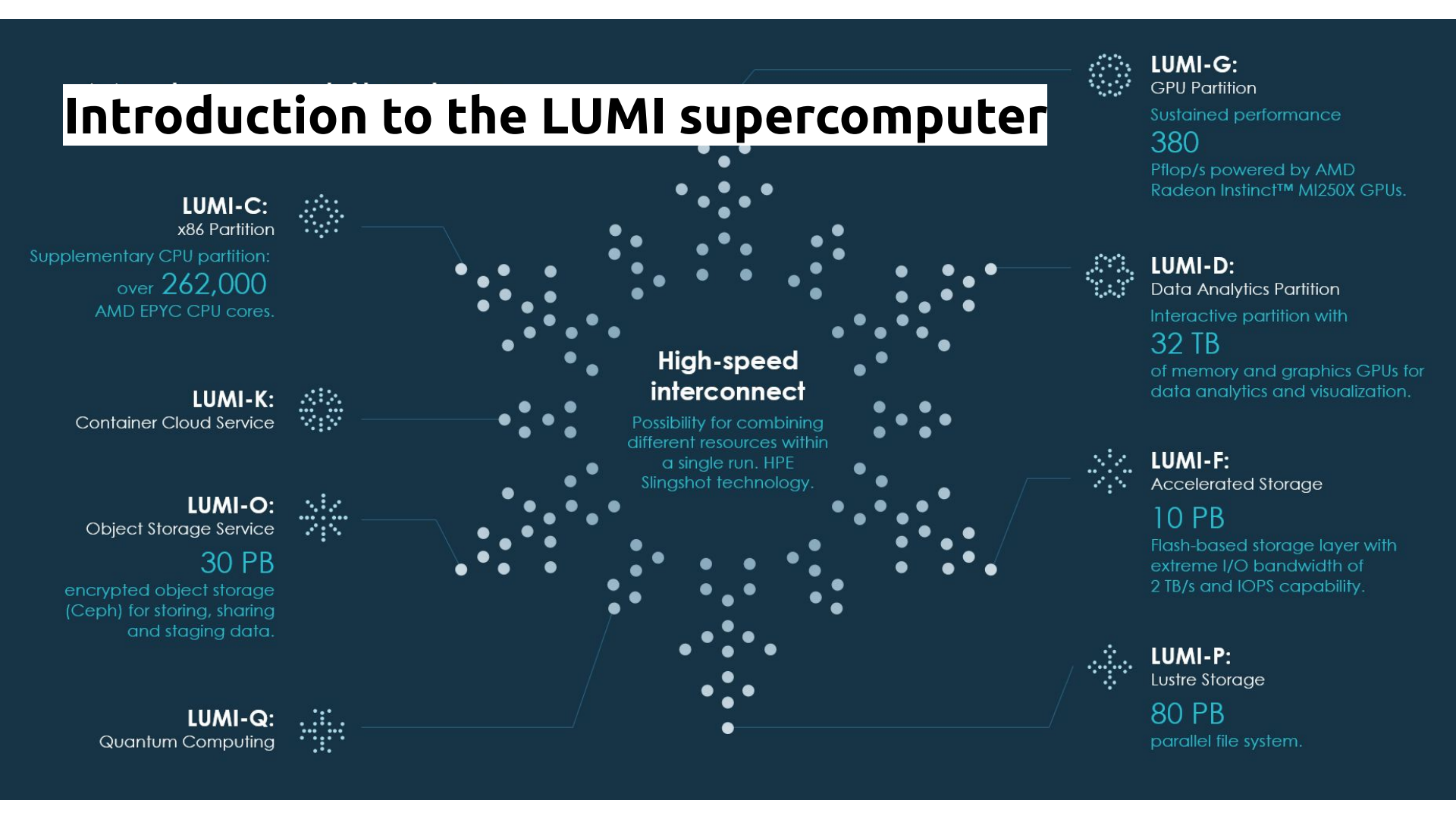
David Samuel and Risto Luukkonen

Introduction to the LUMI supercomputer

Introduction to the LUMI supercomputer

- The fastest supercomputer in Europe, 5th in the world (as of November 2023)
  - 380 petaflop/s in the GPU partition
- It is also the seventh greenest supercomputer on the planet
  - using 100% hydro-powered energy. LUMI's waste heat is used to heat hundreds of households in the city of Kajaani
- Hosted by EuroHPC and the LUMI consortium – Finland, Belgium, the Czech Republic, Denmark, Estonia, Iceland, Norway, Poland, Sweden and Switzerland

Introduction to the LUMI supercomputer



Introduction to the LUMI supercomputer

- 23 824 AMD MI250X GCDs (64 GB each) on 2 978 nodes
- AMD, not NVIDIA with its CUDA and NCCL platforms!
  - Your software has to be compatible with **ROCm** (Radeon Open Compute) and **RCCL** (The ROCm Collective Communication Library)
  - PyTorch officially supports ROCm since version 1.8 — you shouldn't need to change a single line of code to get it working
- Each node has 4x200Gbps Slingshot-11 network interconnects
  - Allows for efficient multi-node distributed training
  - More on scaling later

Case study 1: Training BERT on LUMI

Training a BERT language model on LUMI

- The original BERT-base model can be trained in approximately **6 hours**
- This utilizes the wonderful batch-scaling property of neural-network training
  - Instead of batch size 256 and 1 000 000 optimization steps, we can equivalently use batch size 8 192 and train for 31 250 steps
  - Parallelize over 128 GPUs and you are done in a few hours
- Our models are trained with a custom optimized Pytorch-based codebase: https://github.com/ltgoslo/ltg-bert

75 monolingual BERTs coming soon

- The HPLT project has released new text corpora that cover 75 languages
- LUMI allows us to train a monolingual language model for each of these languages
    - Often the first specialized language model for that language
- Future research: is it better to stay monolingual or go multilingual? How many languages can be combined before we reach the “curse of multilinguality”?

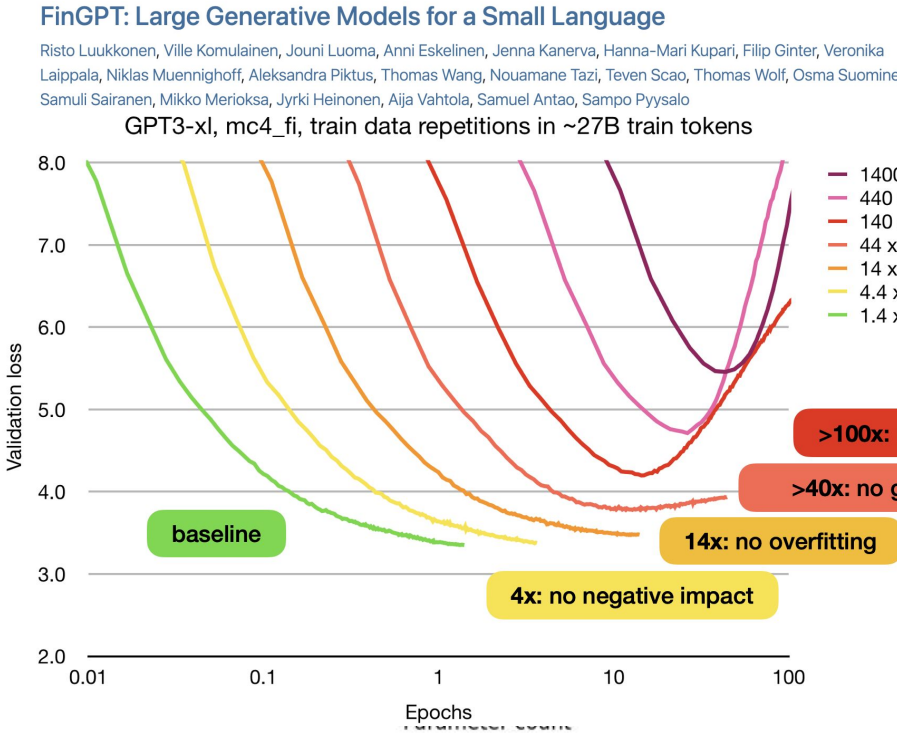
Case study 2: Pretraining Finnish GPTs

FinGPT-project / TurkuNLP

7 monolingual models **180M – 13B (FinGPT)**
1 continued pre-training of **BLOOM 175B**

BLOOM architecture: GPT model with

- ALiBi - positional encoding
- Additional layer normalization after the first embedding layer



Custom Finnish tokenizer with a vocabulary of 131,072 tokens

TURKUNLP .ORG

UNIVERSITY OF TURKU

FinGPT-project / TurkuNLP

7 monolingual models **180M – 13B (FinGPT)**
1 continued pre-training of **BLOOM 175B**

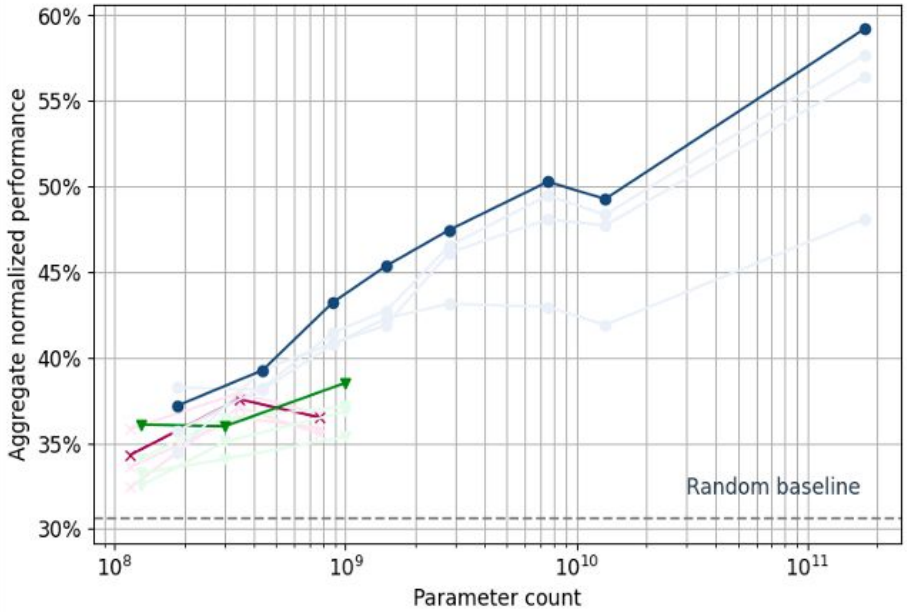
BLOOM architecture: GPT model with

- ALiBi - positional encoding
- Additional layer normalization after the first embedding layer

Custom Finnish tokenizer with a vocabulary of 131,072 tokens

FinGPT: Large Generative Models for a Small Language

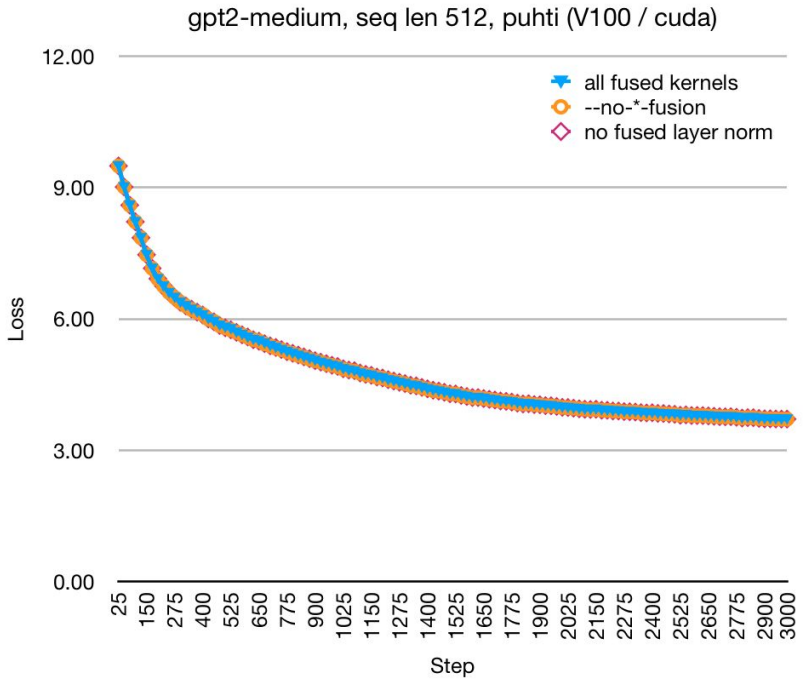
Risto Luukkonen, Ville Komulainen, Jouni Luoma, Anni Eskelinen, Jenna Kanerva, Hanna-Mari Kupari, Filip Ginter, Veronika Laippala, Niklas Muennighoff, Aleksandra Piktus, Thomas Wang, Nouamane Tazi, Teven Scao, Thomas Wolf, Osma Suominen, Samuli Sairanen, Mikko Merioksa, Jyrki Heinonen, Aija Vahtola, Samuel Antao, Sampo Pyysalo



TURKUNLP .ORG

UNIVERSITY OF TURKU

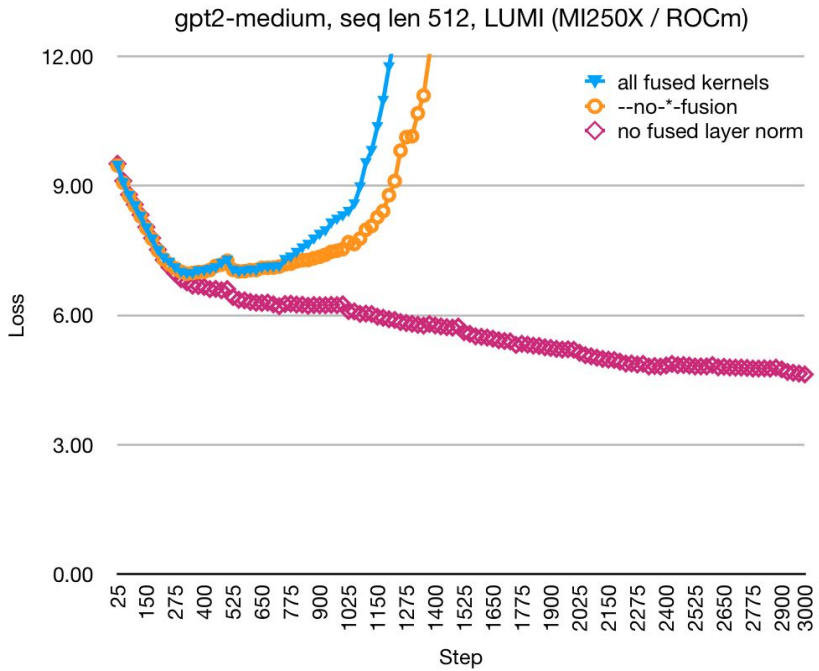
FinGPT-project / TurkuNLP



TURKUNLP .ORG

UNIVERSITY OF TURKU

FinGPT-project / TurkuNLP



Current / future work

1. Can we leverage cross-lingual transfer for small languages?

   → English – Finnish – Code (33B) for 1T tokens

   → Great in Finnish, performant in code, good in English

2. Can we go for a single model for all the Nordic languages?

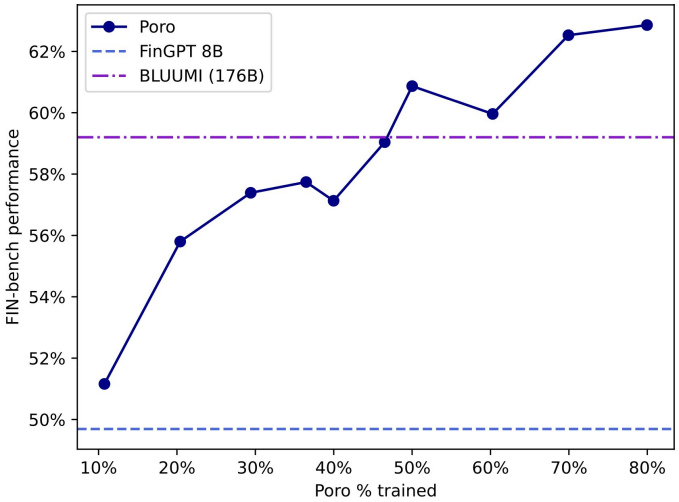
   → English – Finnish – Code + (swe – nor – dan– isl)

   → “Poro - Viking”: (7B, 13B, 33B) for 2T tokens

3. Can we train a performant multilingual European LLM (70B+ params)?

   → Trained for 3-4 T tokens

   Curse of multilinguality?



High Performance Language Technologies

SILOGEN

TURKUNLP .ORG

Pretraining big GPTs on LUMI: requirements

Massive computational budgets

FinGPT – family < 1M GPUh altogether

33B model trained on 1T tokens: ~900k GPUh

Our estimation for a 70-75B param model with 4T tokens: ~8M GPUh with GBS 6.3M tokens

For a 100B model with 4T tokens 8M+ GPUh

Runtime environment for AMD/LUMI

A singularity container with all the software with AMD-specific plugins/patches/kernels (e.g. aws-ofi-rccl-plugin to enable libfabric)

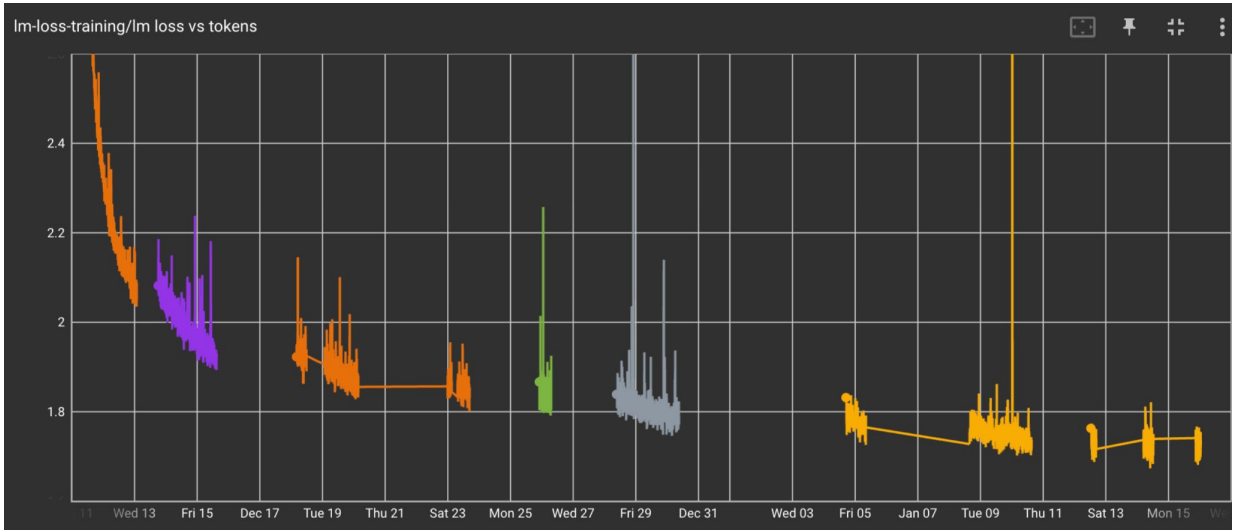
Pretraining big GPTs on LUMI: uptime vs. queue-time

Time-wise expectation management

A shared system
→ LUMI-queues are difficult to predict

Pretraining can take a lot more calendar time as you'd expect

Hardware failures are to be expected

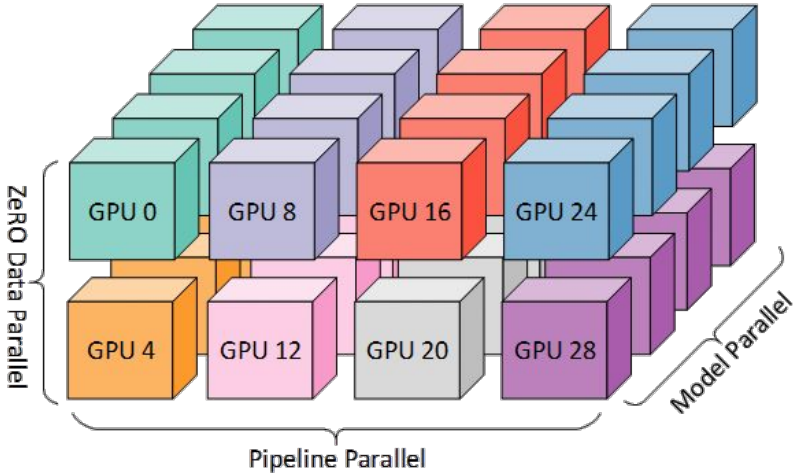


Software – Megatron-DeepSpeed

deepspeed

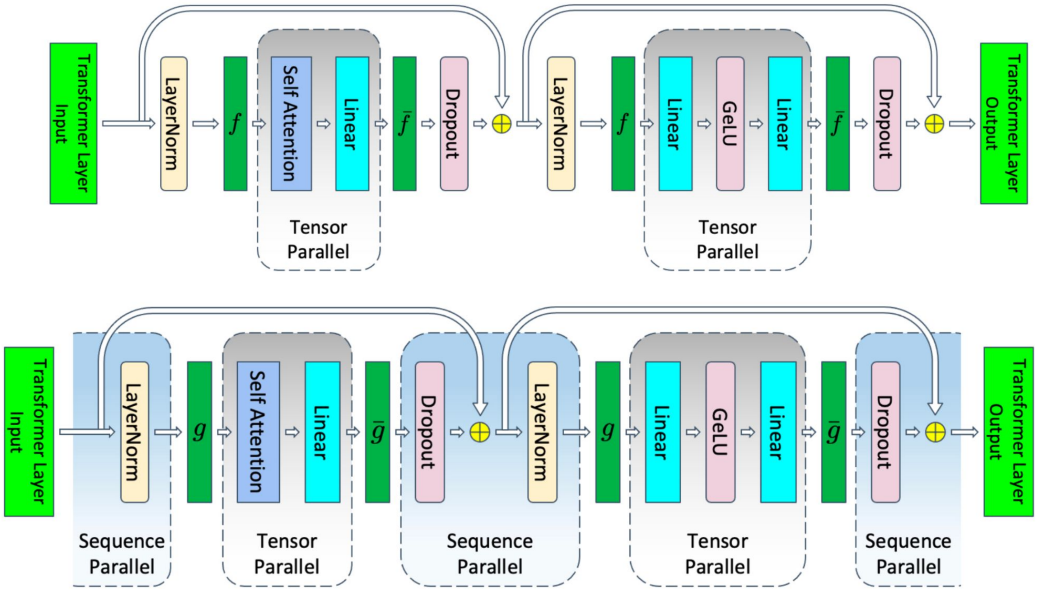
3D parallelism from **Megatron-LM**
- Data parallelism
- Tensor Parallelism
- Pipeline Parallelism

\+ (Sequence Parallelism)



https://huggingface.co/docs/transformers/v4.15.0/parallelism

\+ **Zero redundancy optimizer** from Megatron-DeepSpeed



Reducing Activation Recomputation in Large Transformer Models
(Korthikanti et al. 2022)

Software – Megatron-LM / -DeepSpeed

V1

TurkuNLP/Megatron-DeepSpeed/ **[fingpt-tag]**

Our fork of BigScience/BLOOM

Stable

Had a hipify-related bug on fused-layer-norm.
(Patched with HugginFace and TurkuNLP)

FinGPT-models

Poro 33B

No patched upstream features after 12/2022

V2

microsoft/Megatron-DeepSpeed/

Implements Flash-Attention-support, GQA, sequence parallelism, DeepSpeed-sequence parallelism

Better throughput but more instabilities

Sequence parallelism currently broken on LUMI ❌

V3

Megatron-LM: commit id 4e79e71

Sequence Parallelism works on LUMI ✅

Currently our best throughput for big models with large GBS

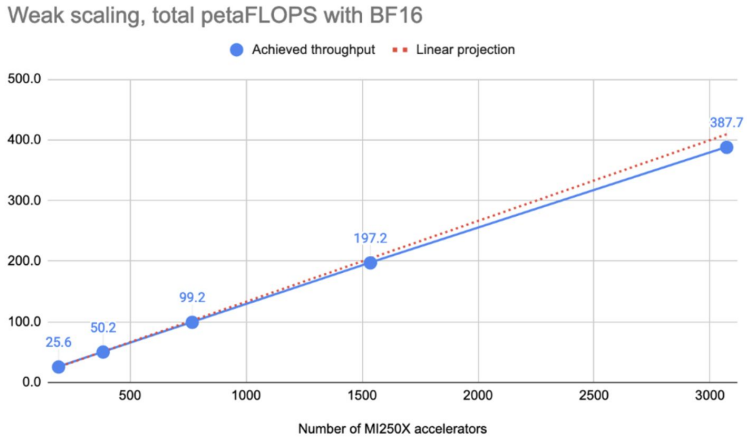
More stable than V2

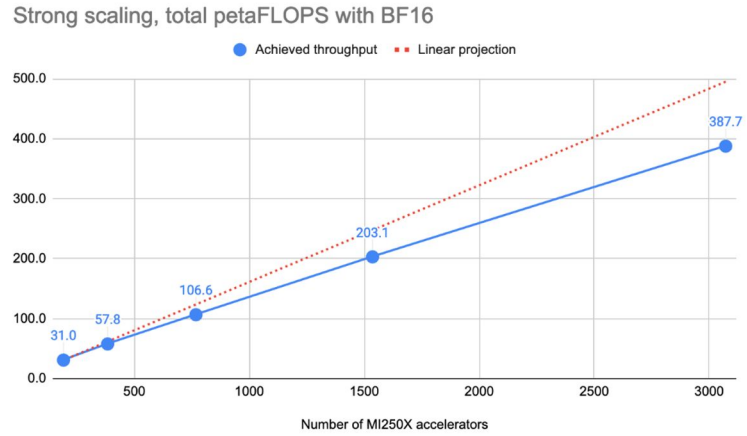
Scaling to thousands of GPUs

Effect of global batch size

- Large → reduced sample efficiency?
- Llama GBS: **4M** tokens
- DeepSeek GBS:
  - **9.4M** for 7B model,
  - **18M** for a 67B model

- Larger GBS → More computation / pipeline stage before synchronization
  → smaller pipeline bubble





DeepSeek LLM Scaling Open-Source Language Models with Longtermism (Bi et al, 2024)

https://www.lumi-supercomputer.eu/scaling-the-pre-training-of-large-language-models-of-100b-parameters-to-thousands-of-amd-mi250x-gpus-on-lumi/

Case study 3:
Pretraining 7B Norwegian LLMs

LLMs for small-ish languages

- What's a good way to pretrain a large language model for a "lower"-resource language?
- Norwegian has about 30B words of available text data
  - That's about 100x less than English!
- Muennighoff et al. (2023): "a few repetitions are okay" → we can get to 120B words or more

Where to find data?

Chinchilla compute-optimality estimates

Parameters	FLOPs	Tokens
400 Million	1.92e+19	8.0 Billion
1 Billion	1.21e+20	20.2 Billion
10 Billion	1.23e+22	205.1 Billion
67 Billion	5.76e+23	1.5 Trillion
175 Billion	3.85e+24	3.7 Trillion
280 Billion	9.90e+24	5.9 Trillion

Data

Compute

LLMs for small-ish languages

- 7B parameters are a sweet spot for a monolingual Norwegian model
- We augment with some code data and repeat the Norwegian texts → 260B tokens
- When parallelized over 512 GCDs on LUMI, training such a model takes less than a week

LLMs for small-ish languages

- 260B tokens is still nothing compared to English models trained on trillions of tokens
- What if we start from a pretrained English model?
  - Then we will inherit a suboptimal tokenizer
- But we can overcome this with two-stage continual pretraining:
  - 1. Train a new tokenizer and train new embeddings for the tokens (freezing the rest of the model)
  - 2. Unfreeze everything and continue pretraining on your corpus
- Very promising results, it substantially outperforms a model pretrained from scratch
  - But what about cultural bias?
  - https://huggingface.co/norallm/normistral-7b-warm