GPT-NL

Sharing insights on scalable & energy-aware LLM pre-training for GPT-NL

Claartje Barkhof & Thomas van Osch

December 12th, 2024

Nederlands Forensisch Instituut
Ministerie van Justitie en Veiligheid

TNO

SURF

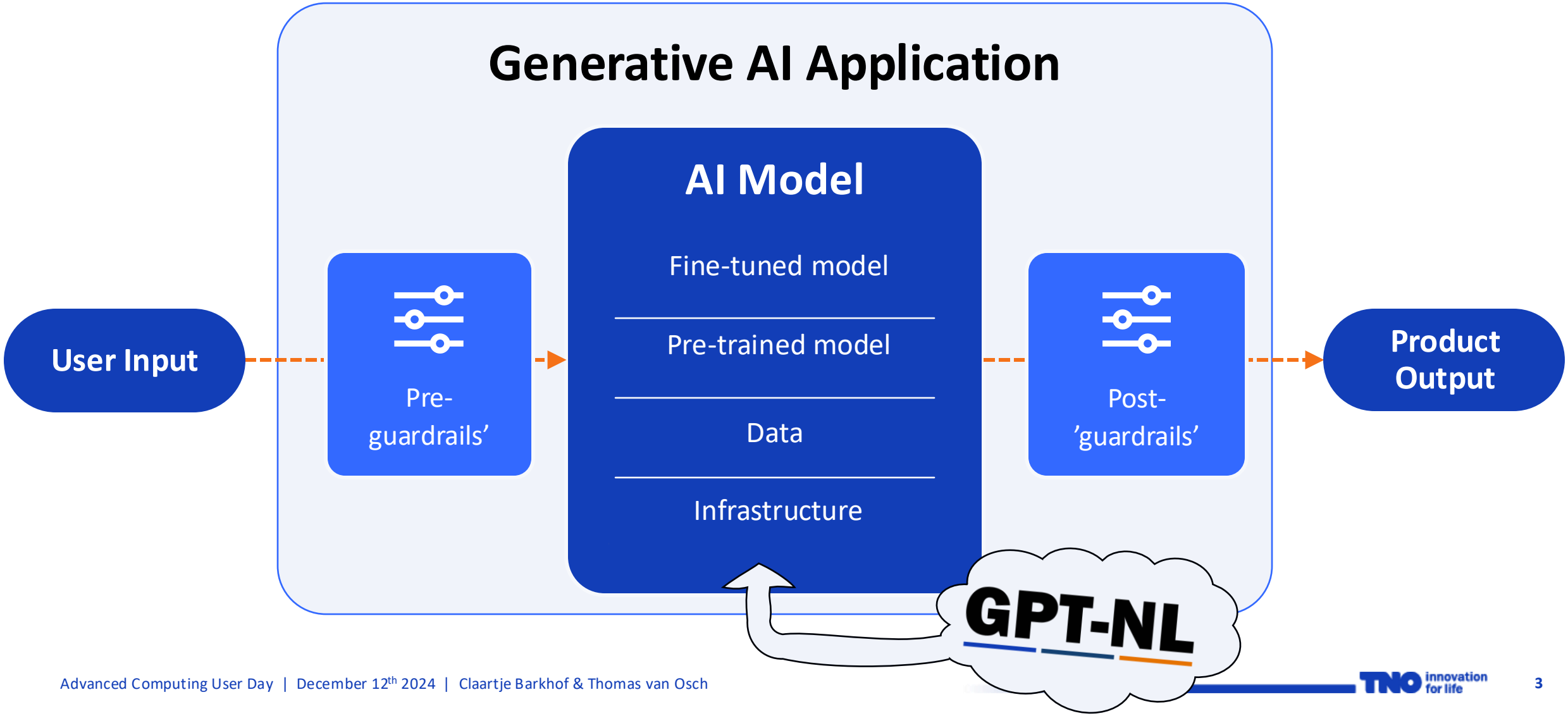
What and why?

- GPT-NL is an LLM that is:
  - Open
  - Transparent
  - Lawful
  - Dutch-centric
  - Sovereign
  - Trained from scratch

Who?

- TNO, SURF, Netherlands Forensics Institute
- Subsidy by EZK, not a research project!

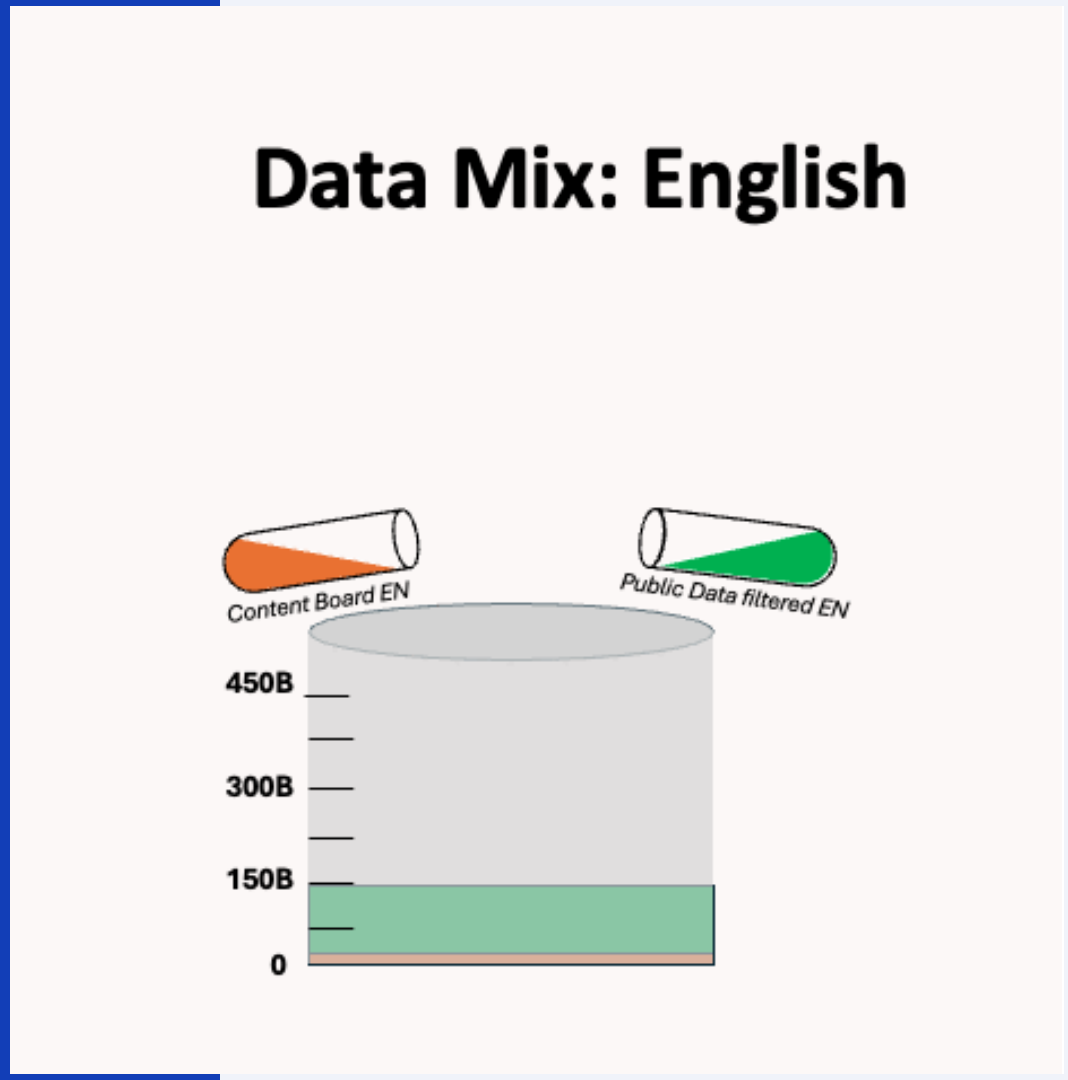
LLM to *enable* applications



Data acquisition

Ethical & lawful are mostly reflected in data acquisition & curation

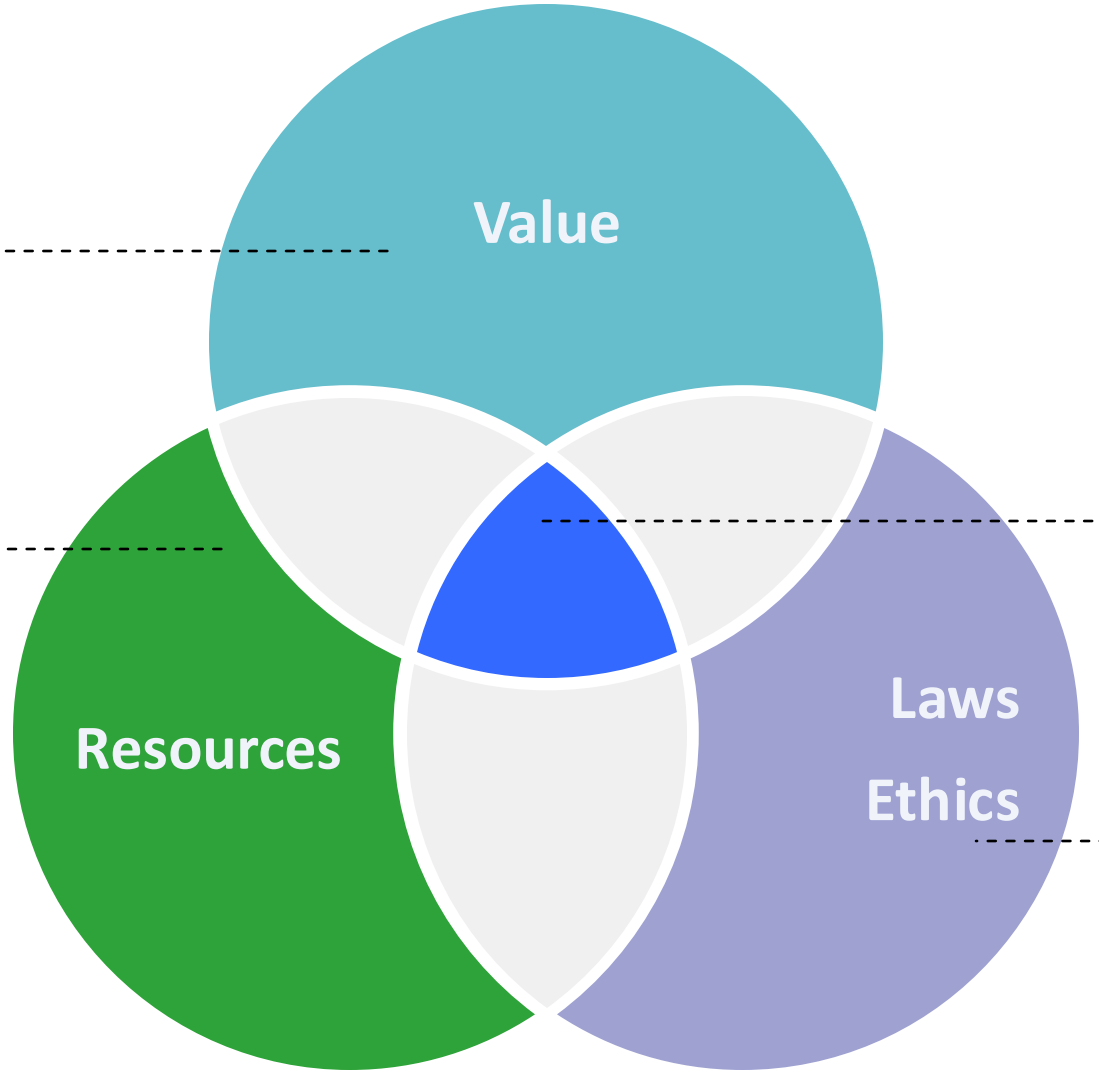
We aim for 450B tokens, based on opt-in and permissive open data



Today's focus

Value
A functional performant model that is in line with our values

Resources
We have limited resources when compared to Big Tech

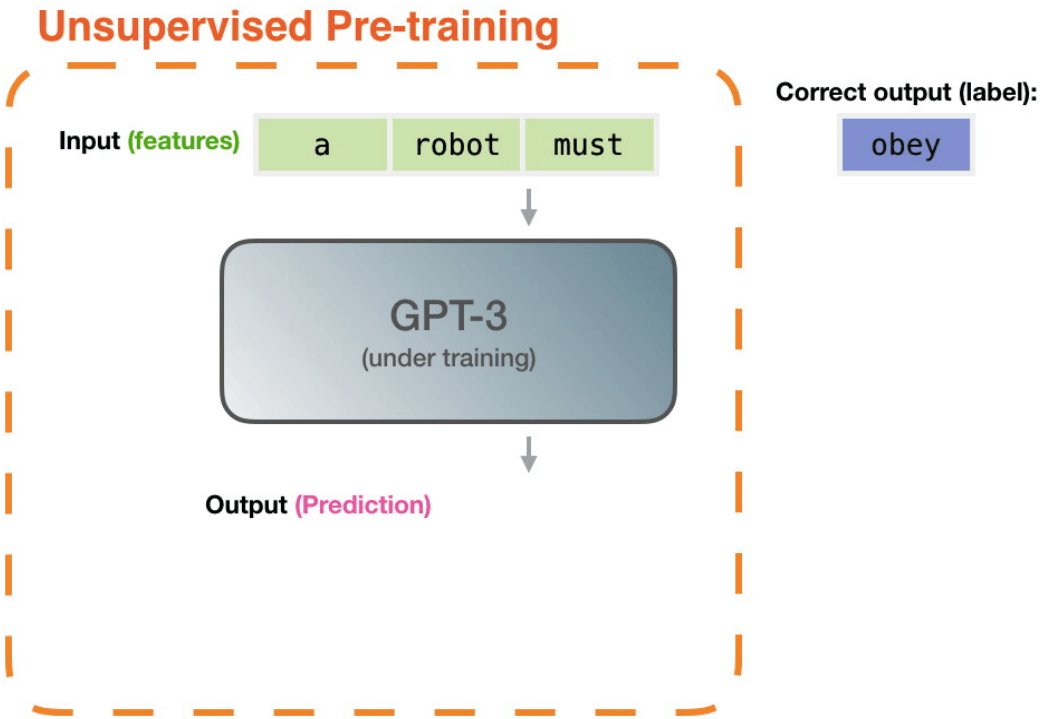


GPT-NL
How can we balance our ambitions with our realistic constraints

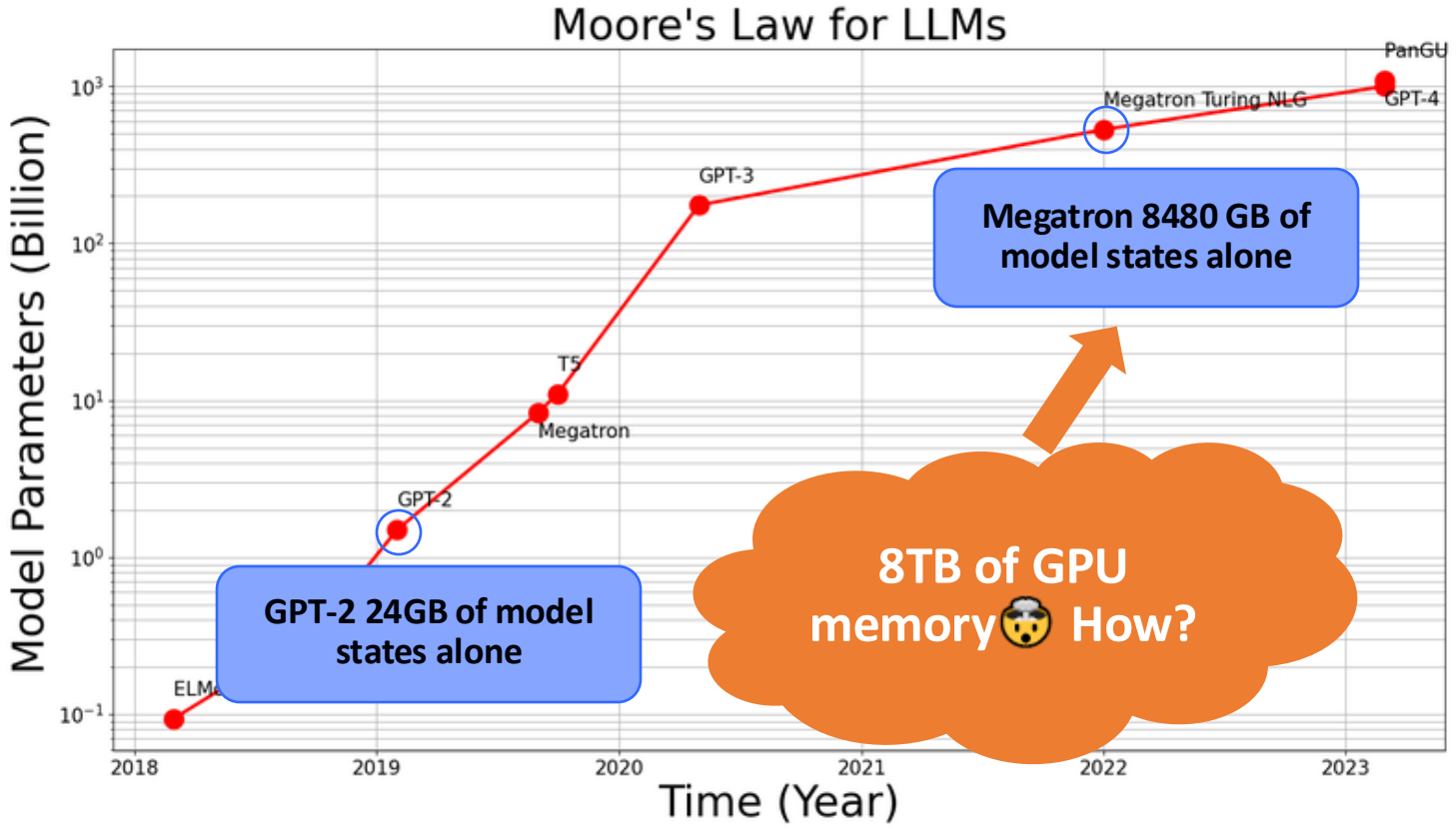
Laws & Ethics
Complying to laws, like GDPR, AI Act and copyright Law.

LLM pre-training on HPC

Relatively consistent pre-training recipe



…while scaling model size



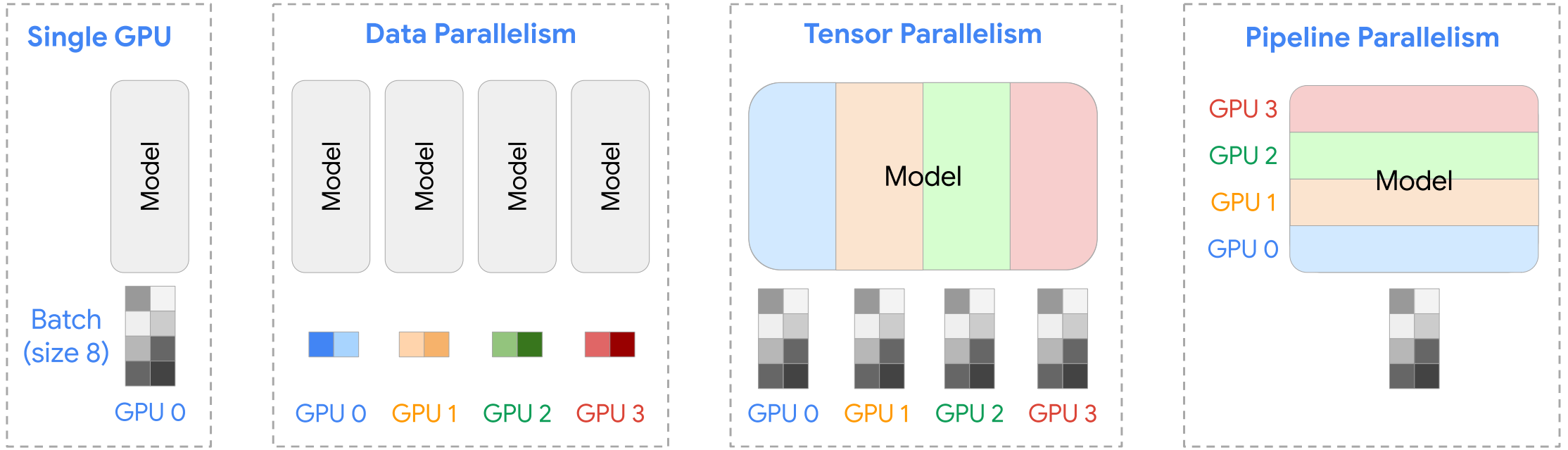
Model parameters
+ Gradients
+ Optimizer states
---------- =
Model states

Distributed training

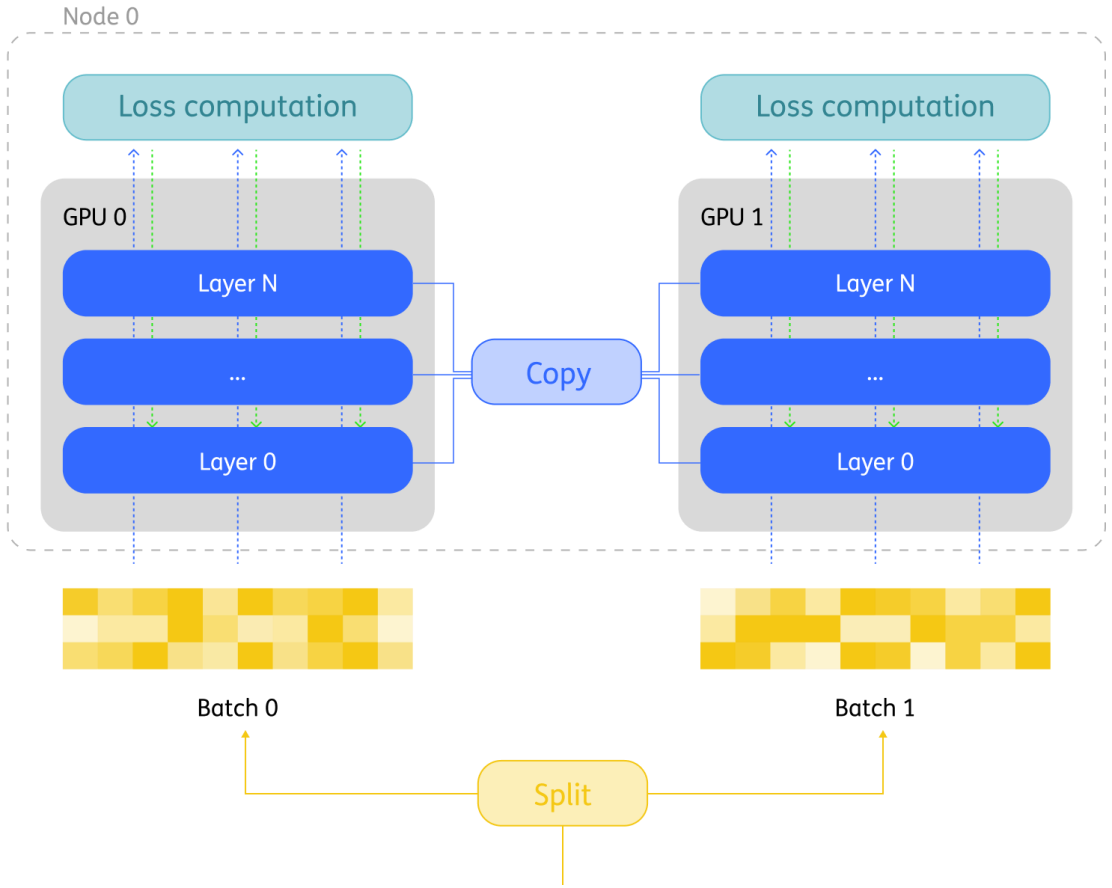
AI HPC

A balancing act between memory efficiency & computational efficiency

How to parallelize training over multiple GPUs (over multiple nodes)?

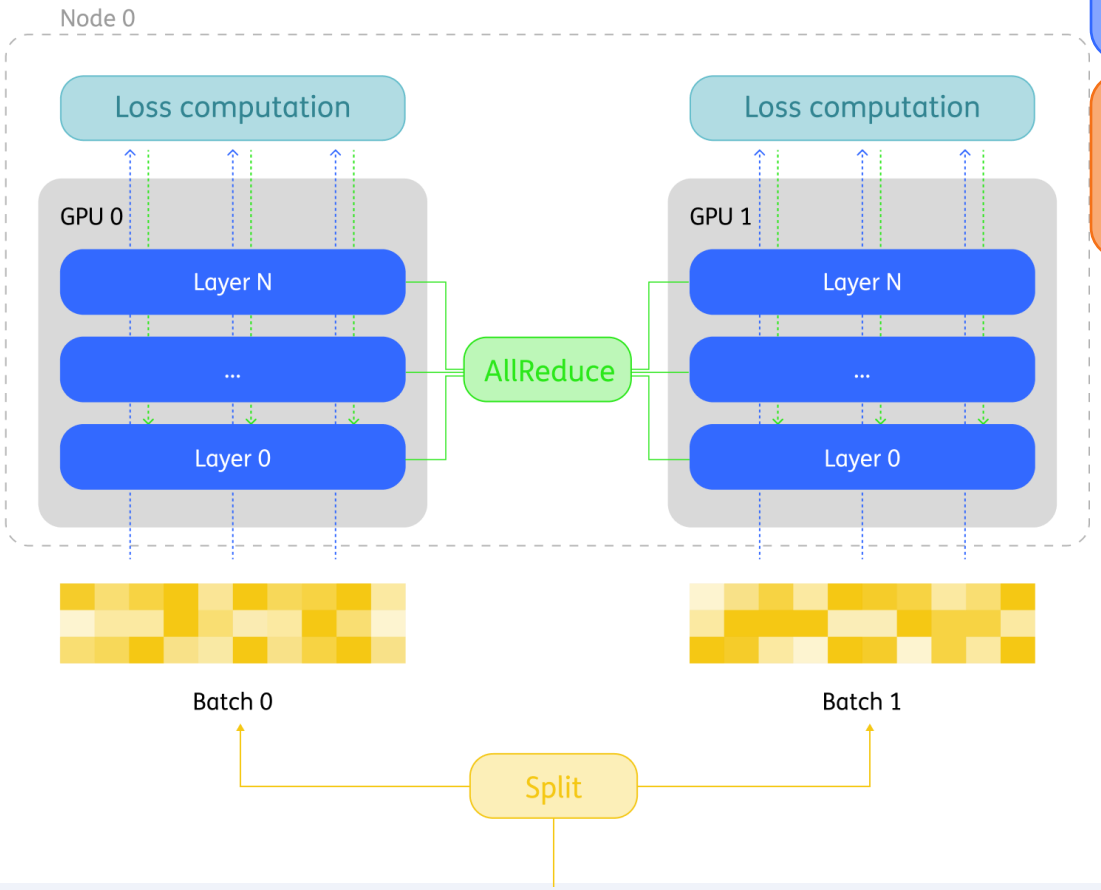


Idea 1: Parallelize the data, copy the weights



Idea 1: Parallelize the data, copy the weights

GPUs only need to synch during the backward pass (average gradients)

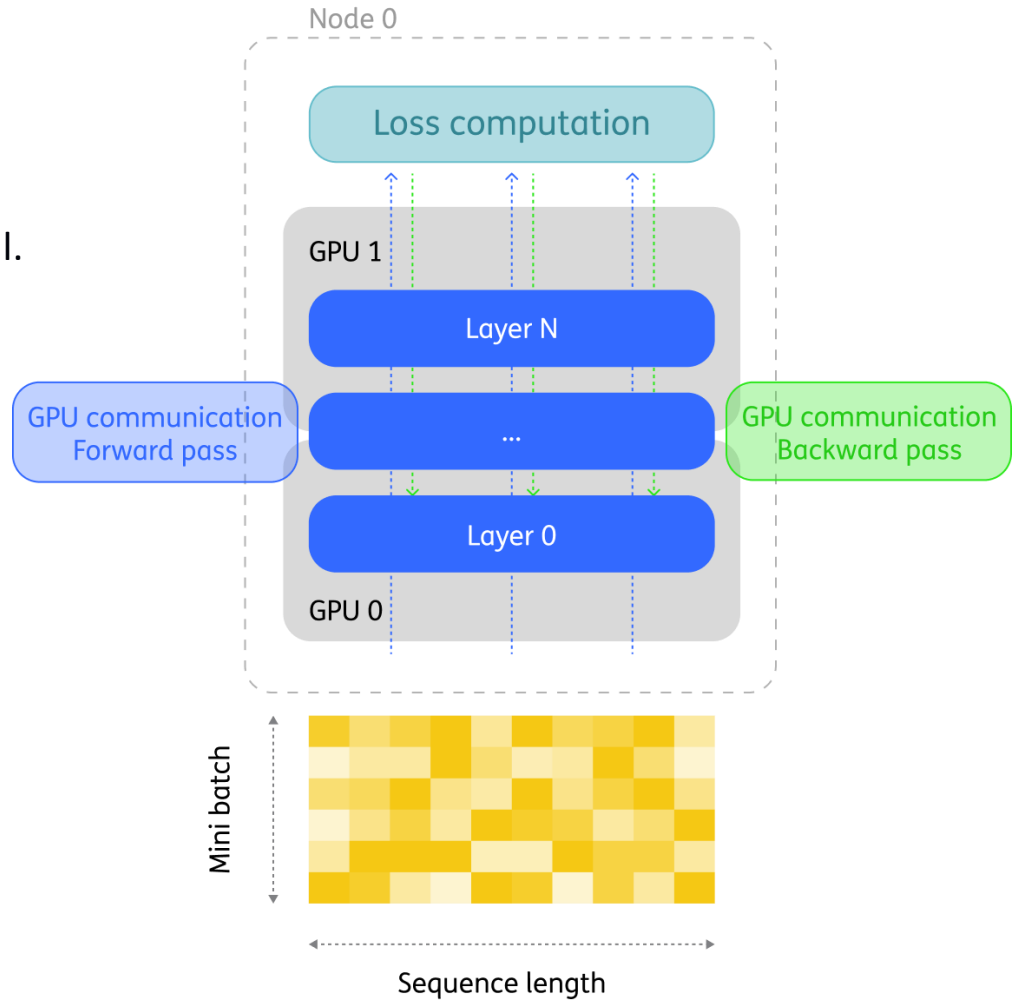


Computationally efficient
leveraging all available hardware to compute things

Memory inefficient
copying weights makes that we take up more space than there is information

Idea 2: Parallelize the model

Instead of distributing the data, we can also distribute the model.



Memory efficient
No weight copying

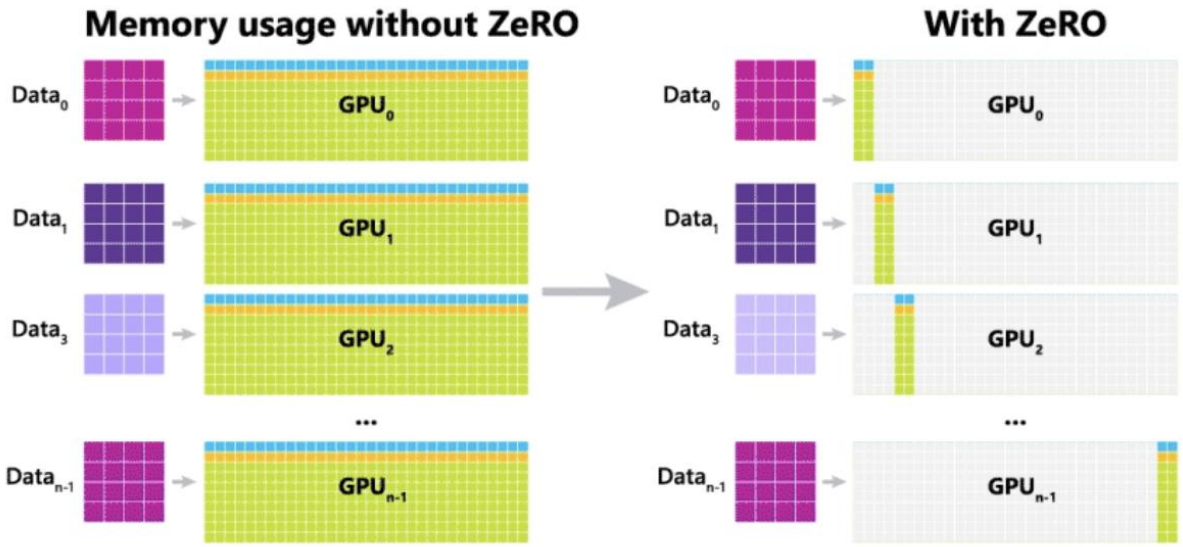
Computationally inefficient
GPUs compute things in turns, leaving them idle for part of the time

Idea 3: use data & model parallelism

- Distribute model states while also parallelizing the data
- Exchange model states between devices when needed

Caveat
There will always be some overhead due to GPU communication

Huge models can now be trained



How to measure GPT-NL training performance?

LLM Compute Calculus

✦450B tokens in context:
± 4.5 million x the first Harry Potter book
± 30 x all Dutch newspapers and magazines

Known

- **Required train compute per token** in FLOPs
  = 6 * number of parameters (or 8 with gradient checkpointing)

- **Theoretical compute budget**
  = time * number of GPUs * TFLOPs (H100 = 989 BFloat-16 TFLOP/sec)

- **Desired number of tokens to train on**
  = 450B (probably oversampled 2-4 times = 900-1800B tokens)✦

Unknown

1. How fast can we process tokens?
2. How efficiently can we make use of Snellius' hardware?
3. How much energy do we consume?

LLM Compute Calculus

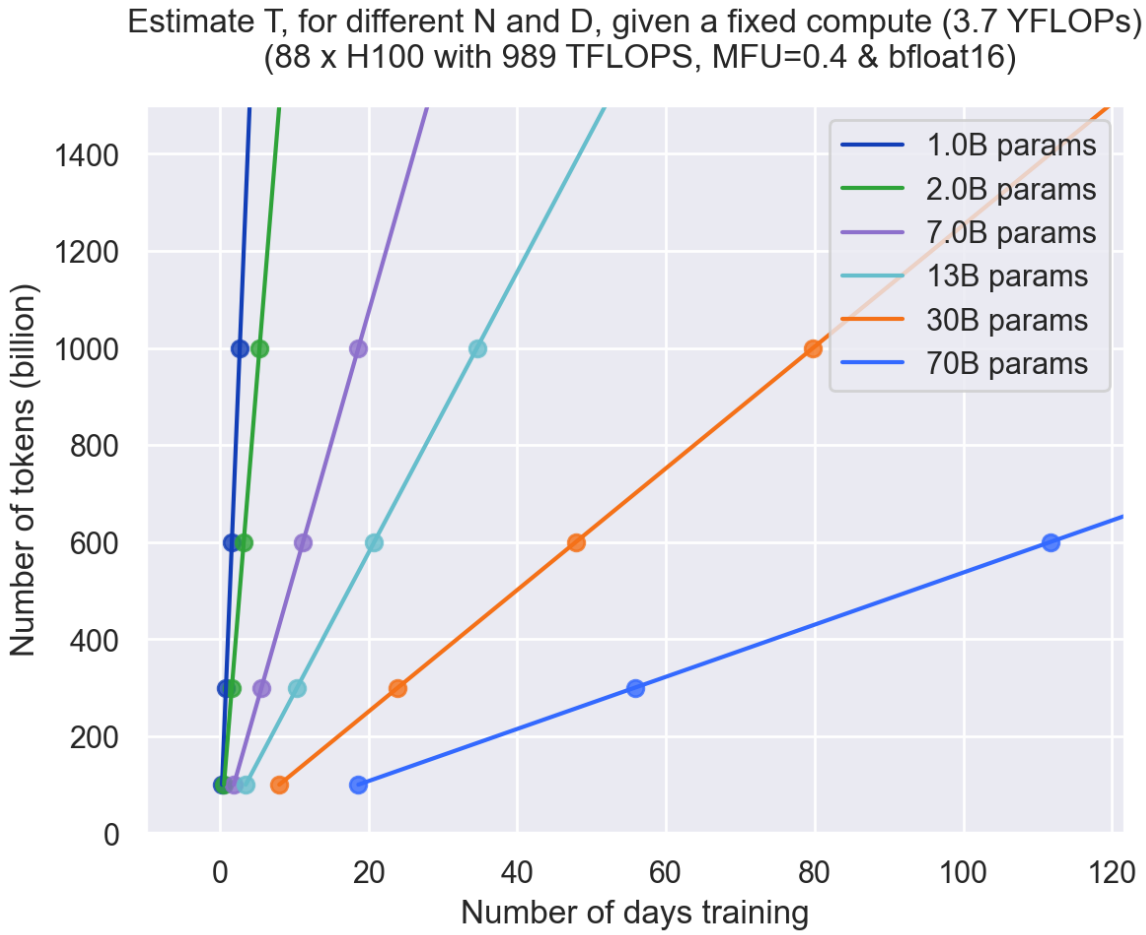
How fast can we process tokens?

Throughput

- In tokens / second
- For a given model size in TFLOP/second

= Actual achieved TFLOP/second

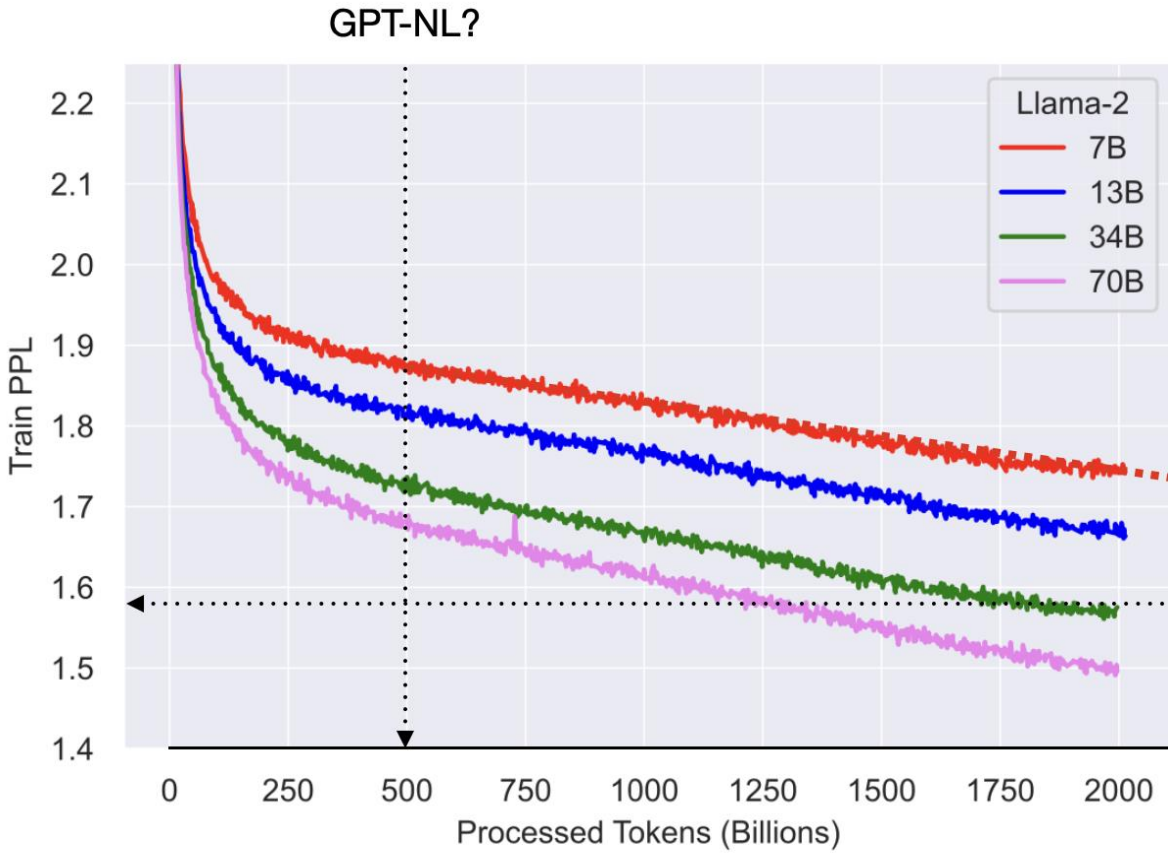
Useful to know how long it takes to train a certain model



LLM Compute Calculus

How fast can we process tokens?

To make educated guesses on the type of model to train



Adapted from the Llama 2 report (Touvron et al. 2023)

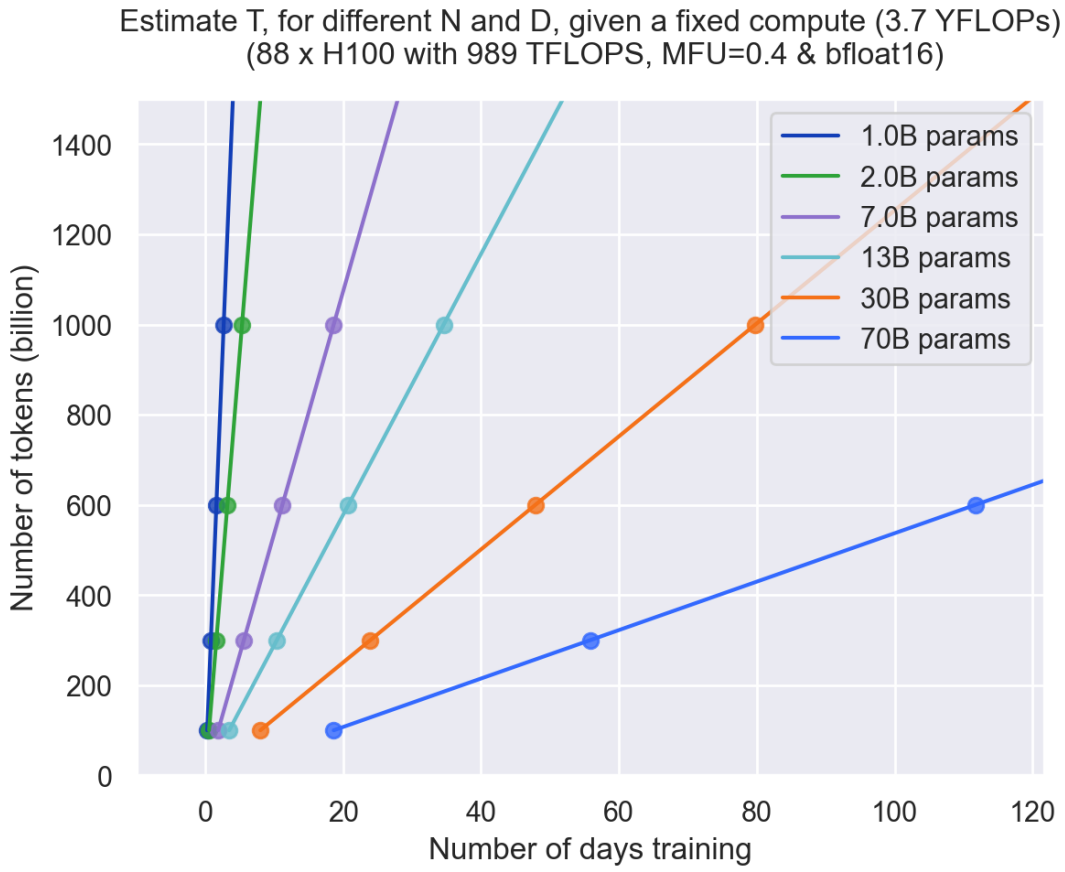
LLM Compute Calculus

How efficiently can we make use of Snellius' hardware?

Model FLOP Utilisation (%)

$$= \frac{\text{Actual achieved Performance}}{\text{Theoretical Peak Performance}} \times 100$$

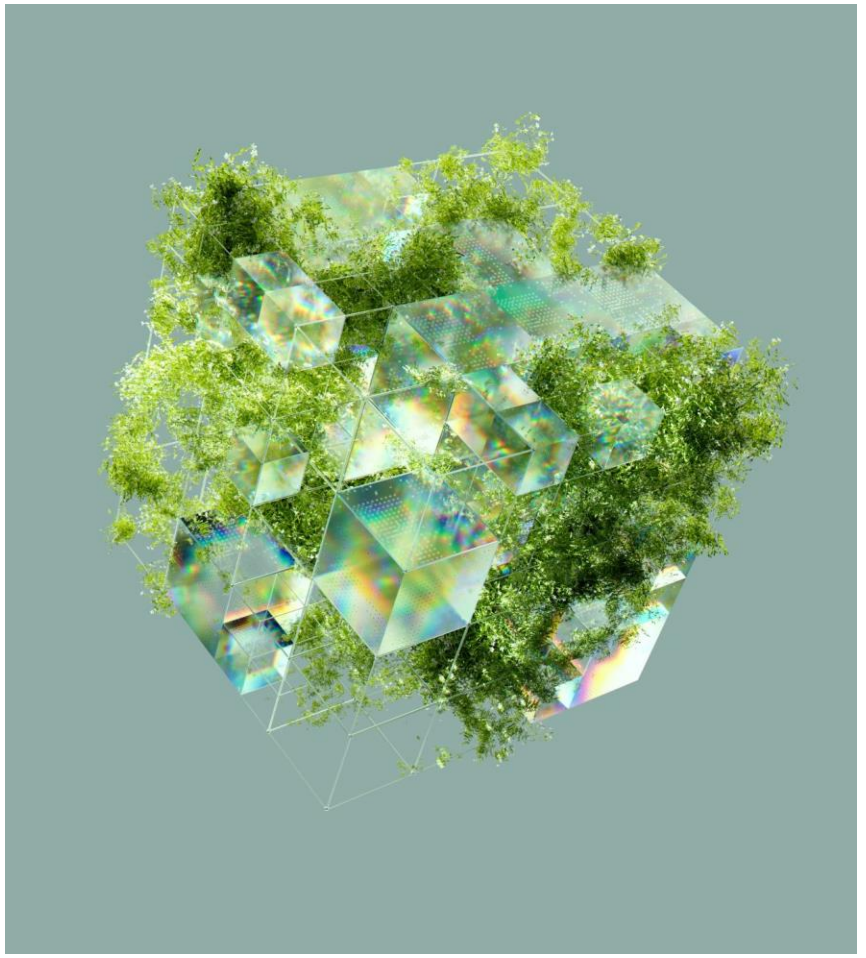
Useful to relate our performance to the hardware



Energy consumption

Why do we care?

- **Transparency**
  - Reporting
- **Making choices with respect to energy consumption**
  - Conscious decisions for training (trade-off model performance versus energy usage)
  - Using automatic optimisations (EAR)

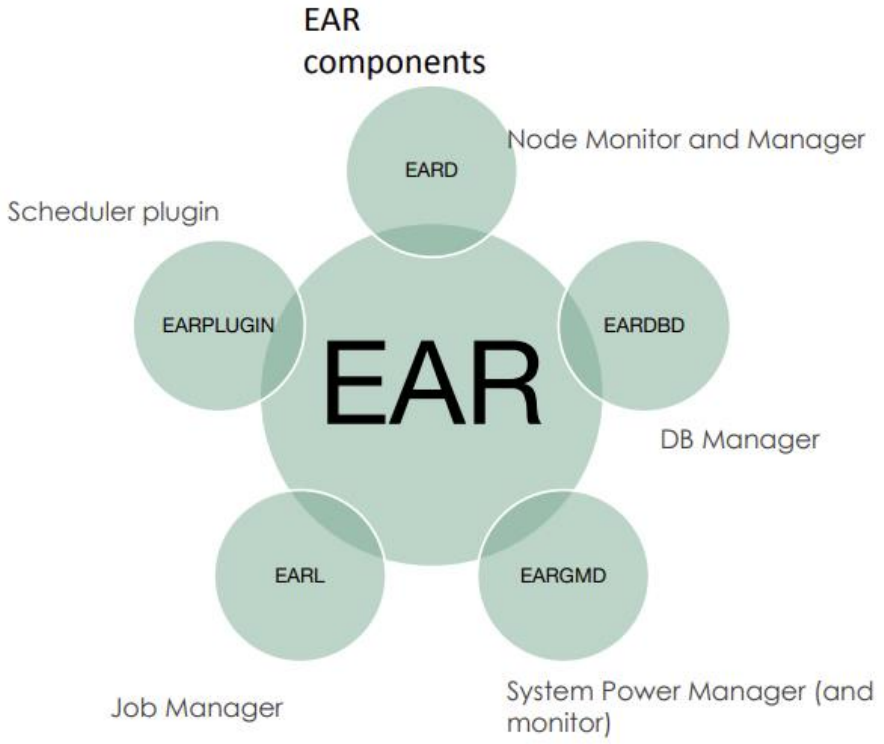


Energy Aware Runtime (EAR)

HPC tooling to measure energy usage

What it can measure:
- CPU energy
- DRAM energy
- Total node energy → very useful

Automatic energy optimization policies



Scaling LLM training on Snellius

Experimental setup 1/2

Evaluation

- Goal: establish scalable pre-training codebase for Snellius
- Measure
  1. Throughput
  2. Hardware efficiency
  3. Energy consumption

Hardware

- 22 nodes with each 4 H100 94GB HBM2e
- InfiniBand network
- Local NVMe storage

Lonely H100 nodes in Snellius

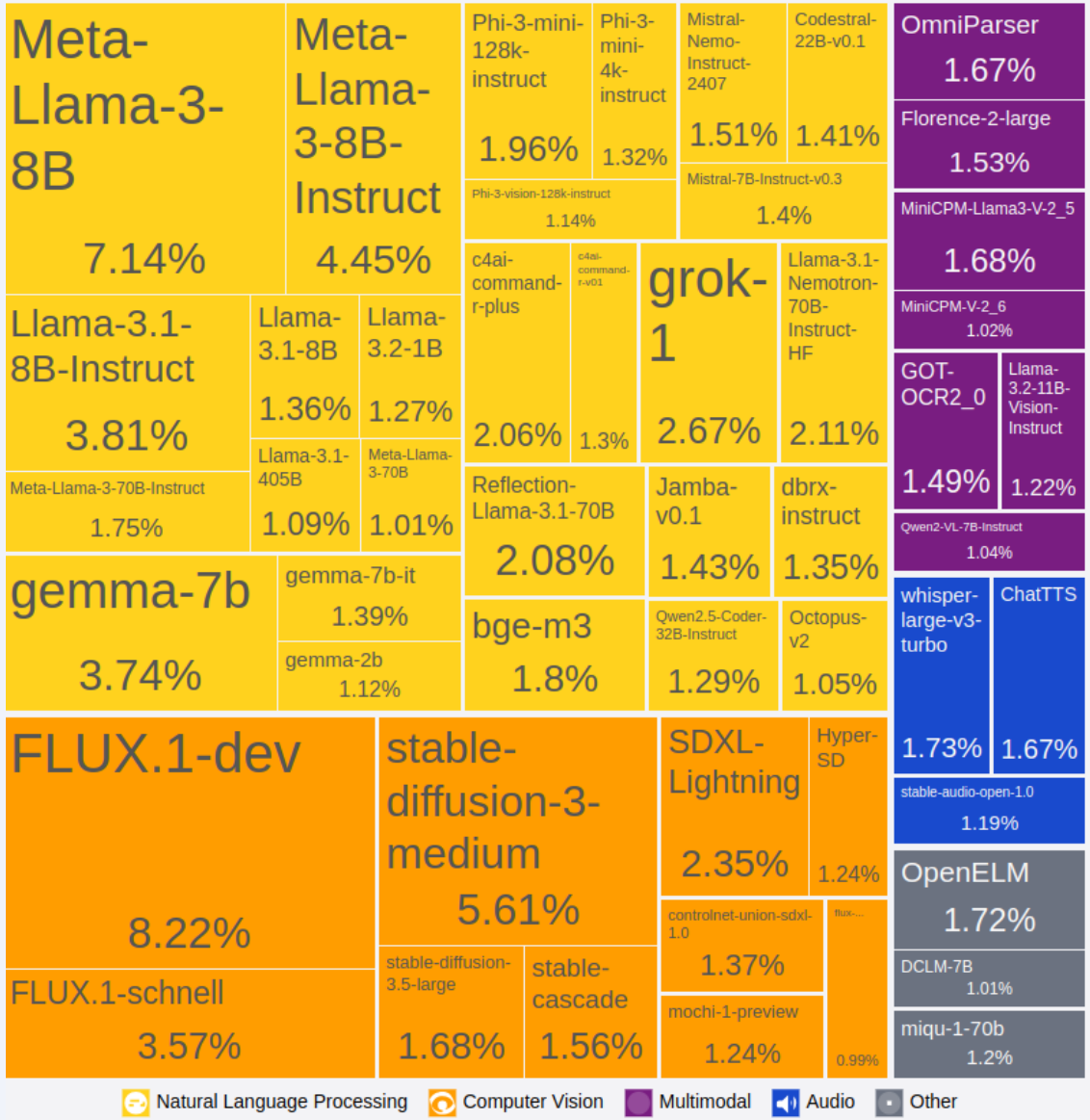
Experimental setup 2/2

Model architecture

- Llama-3 architecture
- 8 billion, 30 billion parameter model
- OLMo's FSDP vs. Hugging Face's Transformers + DeepSpeed

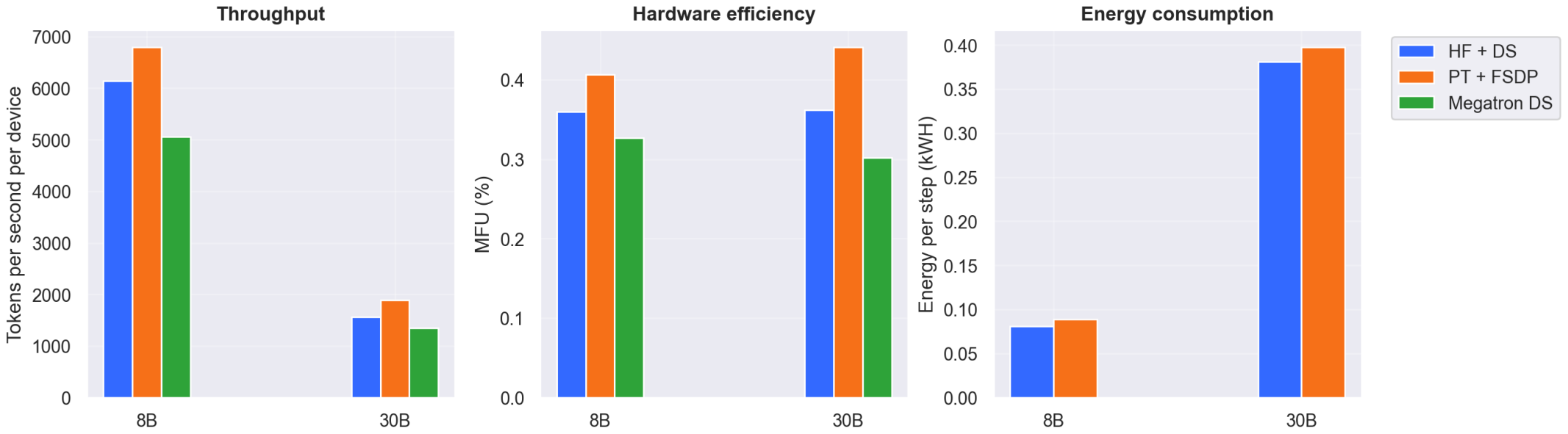
Grid experiment

- 'Small' grid search for parameters on 4 nodes
- Hyperparameter tune:
  - Batch size
  - Sharding strategy
- Strong scaling experiments
- Megatron-DeepSpeed framework as baseline

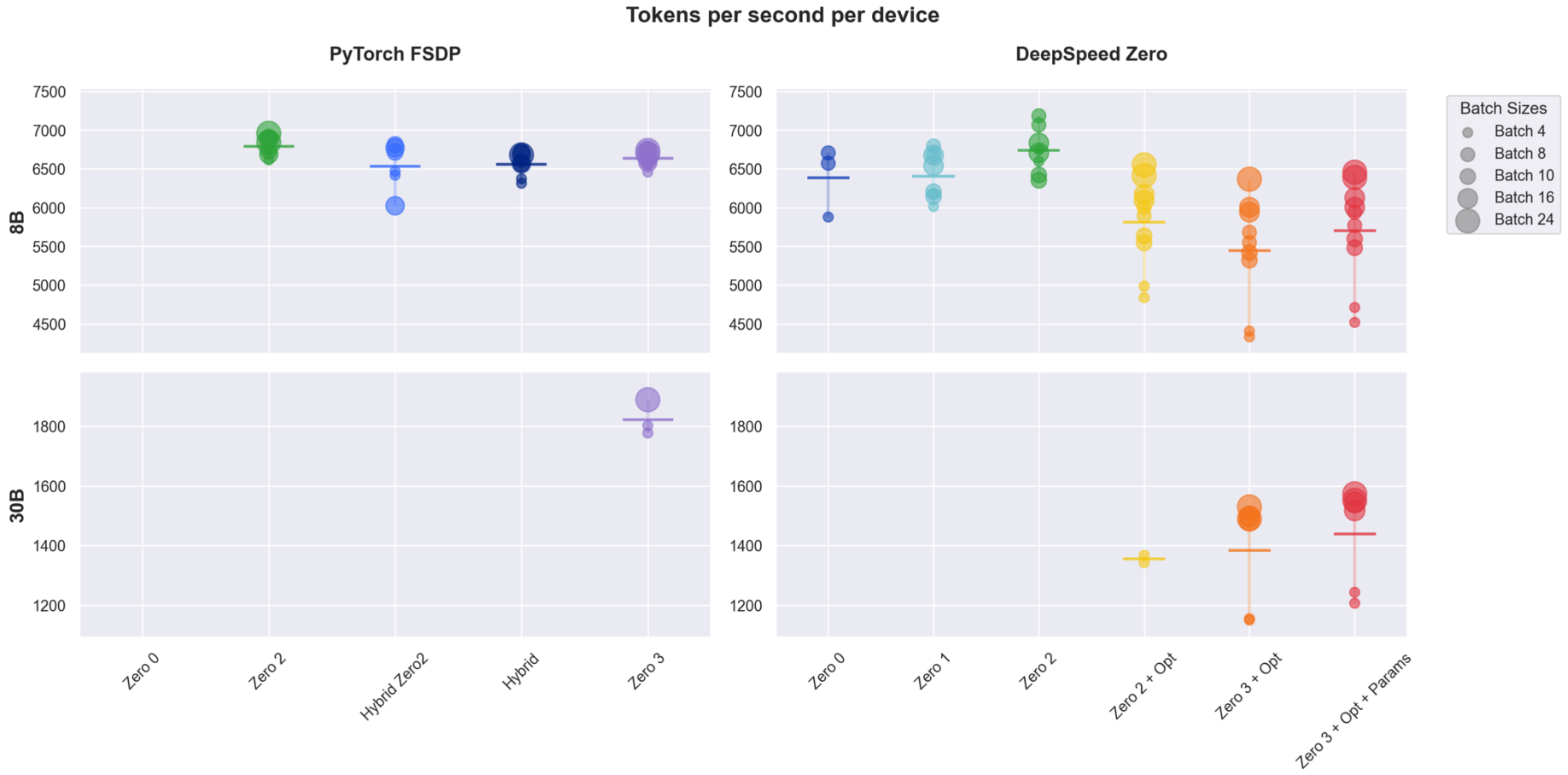


Source: [HuggingFace's open source AI year in review 2024]

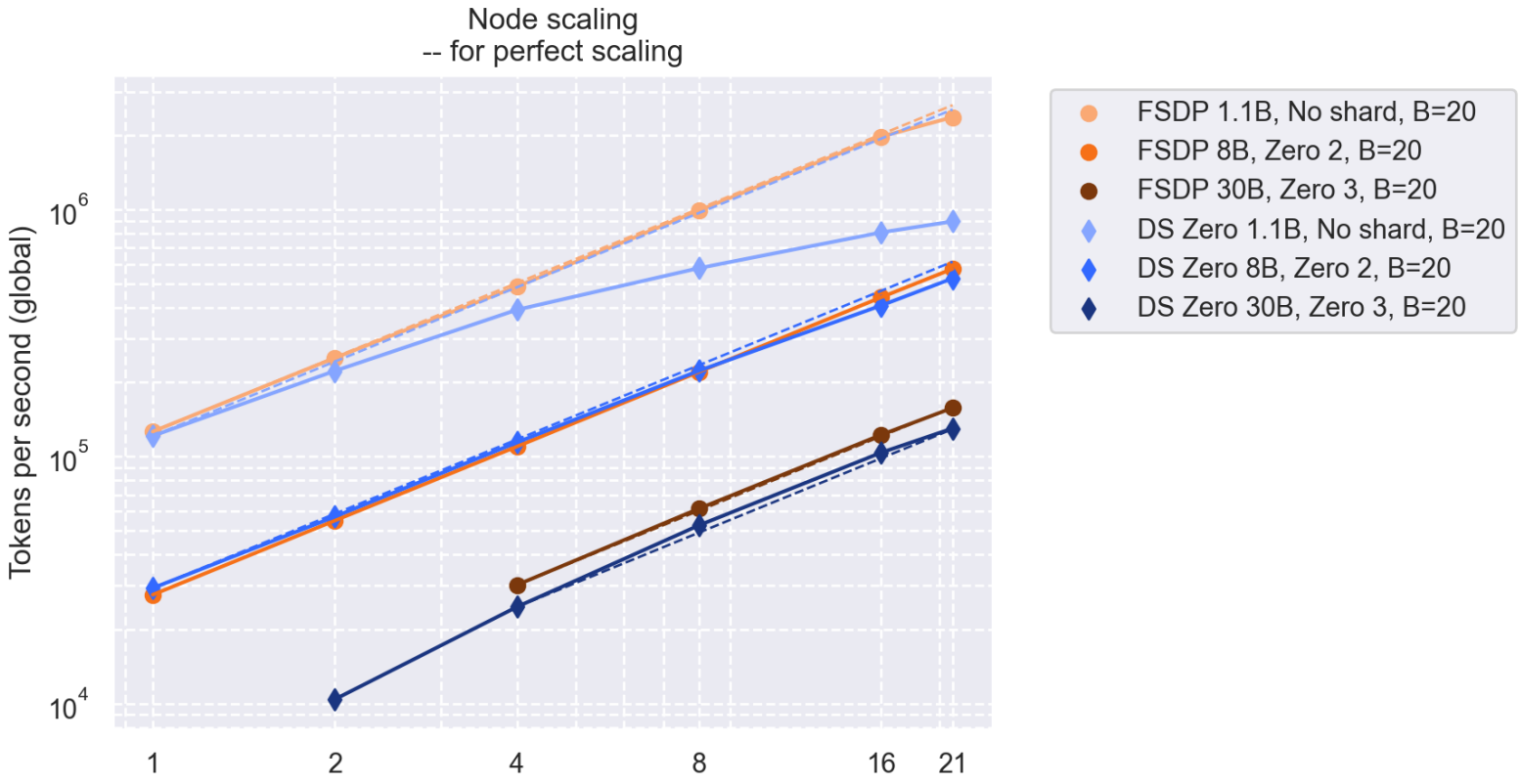
Grid experiment
Best runs (20 nodes)



Comparing sharding strategies



Strong scaling experiment



Technical lessons learned

- **Finetune to your systems hardware**
  - H100s as GPU? Use Flash Attention 3!
  - Fast interconnect network? Use sharding, increase batch size, full precision shards over network!
  - Not a lot of nodes? Don't shard!
- **Scaling LLM pre-training non-trivial**
  - Not your typical HPC problem
  - A lot of knobs to play around **but** hyperparameter tuning not feasible on large scale
- **Community effort and knowledge sharing crucial!**

30B model with 450B tokens equals 34 days training or 14 tonnes of CO2!

Or 40,000 km by car

What's next

- **Goal: establish efficient codebase fit for GPT-NL on Snellius**
- Based on experiments: FSDP better fit for scaling
- Further improve FSDP codebase
  - Fault-tolerance training
- Benchmark **linguistic** model performance
- Start training Q2 2025
- Lunch break!

Thank you!

Credits to the team

Simone van Bruggen
Erik de Graaf
Julio Oliviera
Martino Mensio
Thanasis Trantas
Thomas van Osch
Claartje Barkhof

You can contact us at info@gpt-nl.nl

Additional materials

Sources

- Groeneveld, D., Beltagy, I., Walsh, P., Bhagia, A., Kinney, R., Tafjord, O., ... & Hajishirzi, H. (2024). Olmo: Accelerating the science of language models. arXiv preprint arXiv:2402.00838.
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., ... & Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Practical points

Model

- Pre-training from scratch
- Focus on native Dutch **text**
- Billions-parameter model
- Fundamental use-cases
  - Simplification
  - Summarization
  - Retrieval-Augmented Generation (RAG)
- Not a research project -> publicly funded!

Reporting

- Open-source codebase
- Model weights available
- Document energy and setup

Logistics

- Dedicated share on Snellius
- Training starts Q2 2025

Find suitable codebase for pre-training LLMs on Snellius

PyTorch FSDP

- Adapted from OLMo –
  - open-source repo for pre-training LLM including checkpoints, logging, debugging
- 'Low-level' PyTorch
- Flexibility
- Support for FSDP (Fully Sharded Data Parallel)
- Established codebase

Hugging Face's Transformers & DeepSpeed

- Seamless integration with external tools (logging, visualization, etc.)
- More boilerplate code
- Excellent versioning and future-proof
- Support for DeepSpeed ZeRO
- Established codebase (more for finetuning)

Llama 3 architecture

Llama3 Transformer architecture with following key hyperparameters:

	8B	30B
Layers	32	60
Model dimension	4096	6656
FFN dimension	14336	17920
Attention heads	32	52
Key-value heads	8	13
Attention	Grouped query attention	
Positional embeddings	RoPE (ϑ = 500,000)	
Memory efficiency tricks	KV cache	
Activation	SwiGLU	

Grid experiment

	A	B	C	D	E
1	Batch Size	Flash Attention	PyTorch (PT)	HuggingFace (HF)	Equivalent?
2	4,8,10,16,24	On/Off	NO_SHARD	Zero 0	✓
3	4,8,10,16,24	On/Off	-	Zero 1	HF only
4	4,8,10,16,24	On/Off	SHARD_GRAD_OP	Zero 2	✓
5	4,8,10,16,24	On/Off	HYBRID_SHARD_ZERO2	-	PT only
6	4,8,10,16,24	On/Off	-	Zero 2 + CPU offload opt	HF only
7	4,8,10,16,24	On/Off	HYBRID_SHARD	-	PT only
8	4,8,10,16,24	On/Off	FULL_SHARD	-	PT only
9	4,8,10,16,24	On/Off	-	Zero 3 + CPU offload optimizer	HF only
10	4,8,10,16,24	On/Off	-	Zero 3 + CPU offload optimizer & params	HF only

ZeRO 4-way data parallel training

Using:

- P_{os} (Optimizer state)
- P_g (Gradient)
- P_p (Parameters)