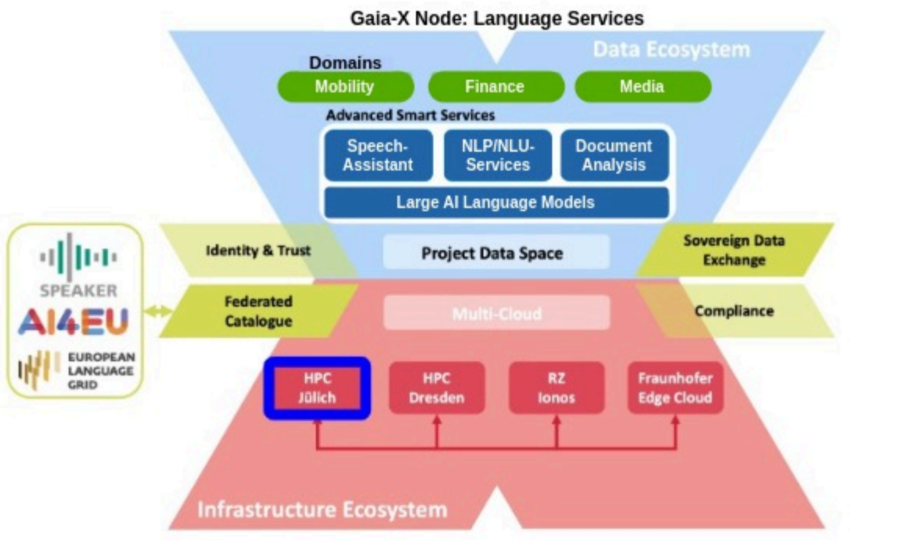
# OpenGPT-X: Training Large Language Models on HPC Systems

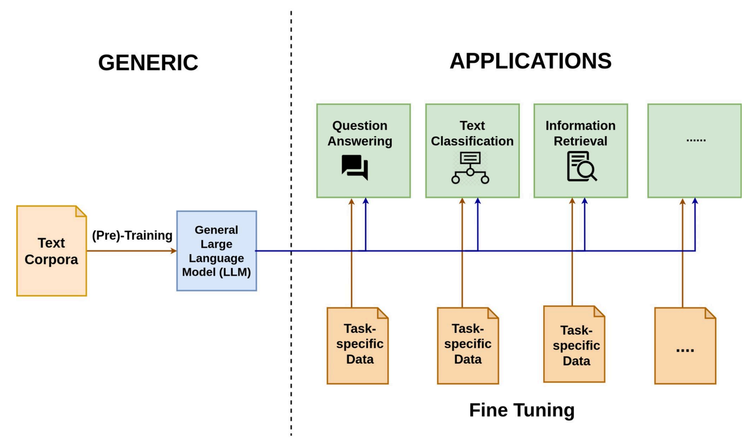
Chelsea John, Jan Ebert, Stefan Kesselheim, Carolin Penke, Andreas Herten

## OpenGPT-X (2022 - 2024)

- German initiative to **build and train large-scale AI language** models for innovative language applications
- **Commercialization** through Gaia-X infrastructure for **European Economy**
- Consortium of 11 partners from **industry and academia**




## Large Language Models (LLM)




**LLM Transformer Architecture:** Uses stack of **encoders/decoders** to process data, weighted by **attention**.

**Use Cases:**
- Customer service chatbots
- Translation, autocorrect, autocompletion
- Document summarization and generation
- Insurance claims management
- Fraud detection

**Language Models:**
- **BERT** *(2018)*; by Google
- **GPT** models, GPT-1 *(2018)*, GPT-2 *(2019)*, GPT-3 *(2020)*, GPT-4 *(2023)*; by OpenAI
- **OPT** *(2022)*, by Meta
- **BLOOM** *(2022)*; by BigScience *(HuggingFace)*; based on Megatron-Deepspeed
- **OGPT-1(?)**, by OpenGPT-X, TBD

## Model Training

Key Technique: **Parallelization** (memory constraints, large input data)

**1. Distributed Data Parallelism (DP):**
- Full model on each rank
- **Training data distributed** in micro-batches
- Gradients averaged across all ranks via *allreduce*

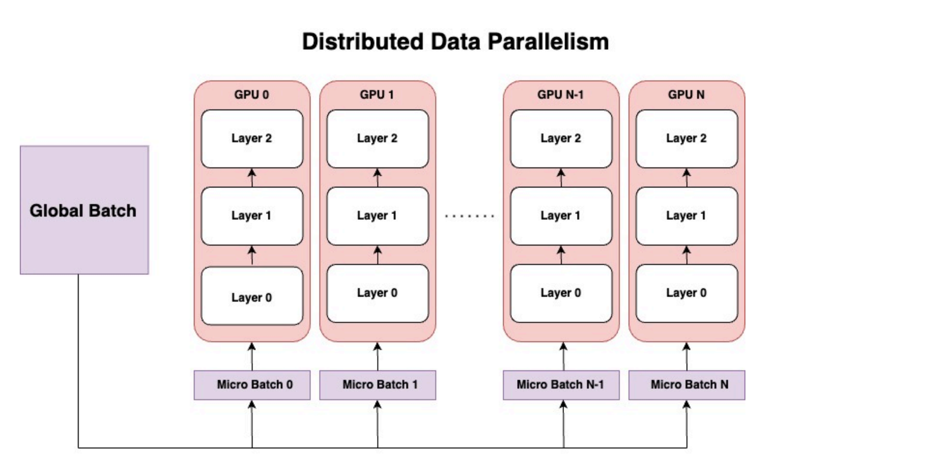
$$\text{Global BatchSize} = \#DP \times \text{Micro BatchSize}$$

**2. Pipeline Parallelism (PP):**
- **Model layers partitioned** across ranks (*vertically*)
- Asynchronous pipe scheduling for gradient accumulation and calculation

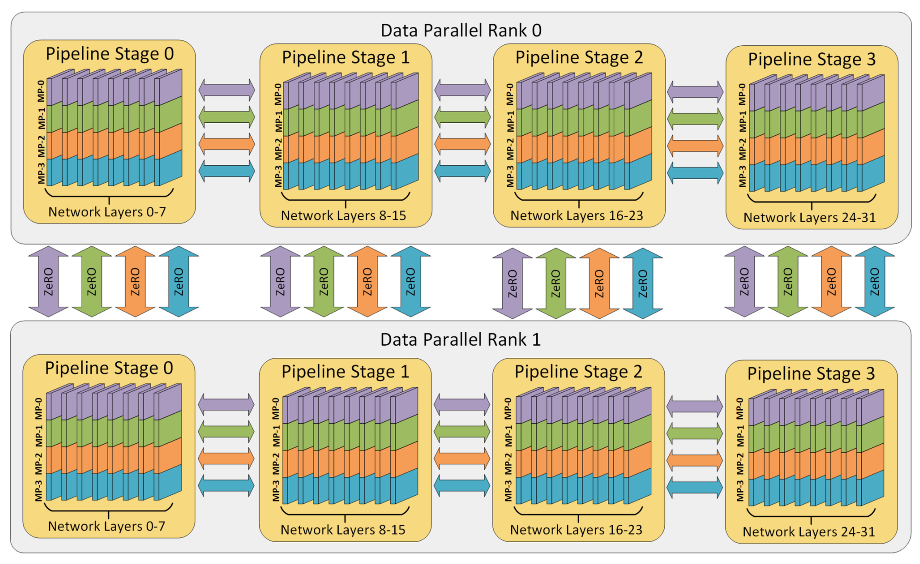
**3. Tensor Parallelism (TP):**
- **Tensor operations partitioned** across ranks (*horizontally*)
- Communication-intensive with frequent *allreduce*

⟶ *Use all 3 levels to determine number of tasks / number of GPUs*

$$\#GPUs = \#DP \times \underbrace{\#PP \times \#TP}_{\#MP} = \#DP \times \#MP$$




Example for Distributed Data Parallelism: entire model fits on single GPU. Model processes data in units of micro batches, performs model update with averaged gradients.

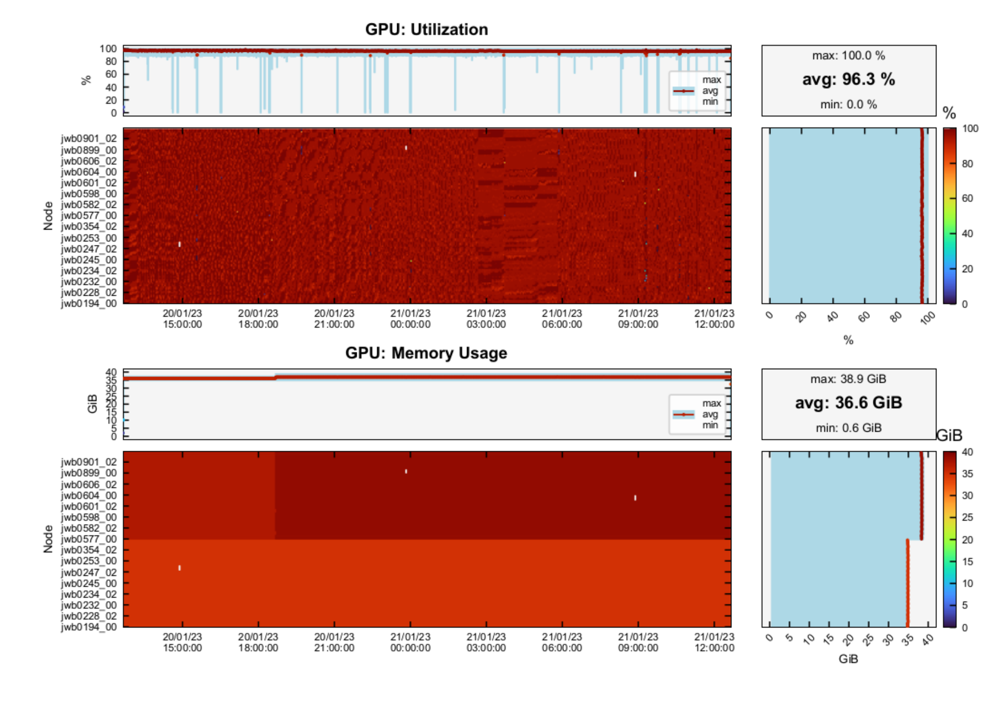



Example of different parallelization schemes. MP (Model Parallel) was reformulated to TP, with MP=PPxTP.
Parallelization using DeepSpeed ZeRO, https://www.deepspeed.ai/tutorials

## Training 13.6B Model on JUWELS Booster

- Basis: Megatron-DeepSpeed (fork)[1]
- 13.6B Model: 13.6 Billion parameters
- Size: 56 GB (Parameters + Gradients + Optimizer states, *ZeRO Stage 1*[2])
- Partition: #PP=2, to fit 40 GB A100 GPU (→28 GB per GPU)
- Scaling: #DP=80
- Training on German-English data with GlobalBatchSize=960, MicroBatchSize=2 and GradientAccumulationStep=6
- 160 GPUs (40 nodes) on **JUWELS Booster**[3]

*[1]: Private repository forked from https://github.com/bigscience-workshop/Megatron-DeepSpeed*
*[2]: ZeRO: Memory Optimizations Toward Training Trillion Parameter Models; arXiv:1910.02054 [cs.LG]*
*[3]: JUWELS Booster: > 3200 Nvidia A100 GPUs, 40 GB; https://apps.fz-juelich.de/jsc/hps/juwels/booster-overview.html*
*[4]: https://www.fz-juelich.de/en/ias/jsc/services/user-support/jsc-software-tools/llview*

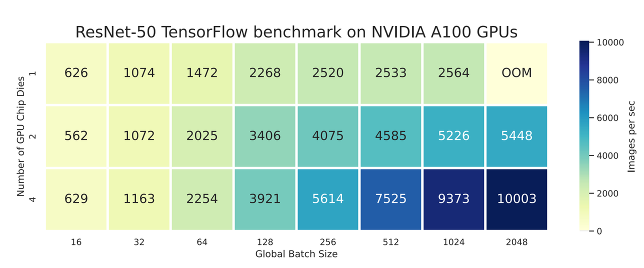



Report generated using LLview[4]. Training of 13.6B model (TP=1, PP=2 and DP=80): GPU utilisation of 96.3% (avg) making use of 36.6 GB (avg) memory.

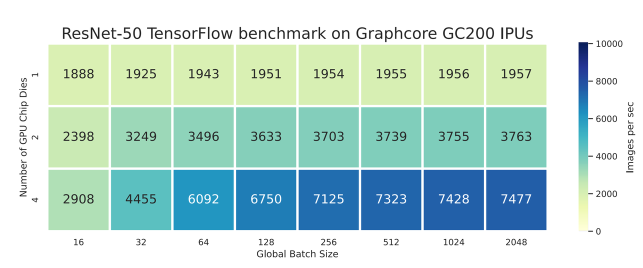

Tensorboard: Tokens vs. Training Loss Plot; for 13.6B model trained on 12 billion tokens, 160 GPUs, 24 h. Decreasing loss attests for model quality and convergence.

## Novel Architecture Exploration

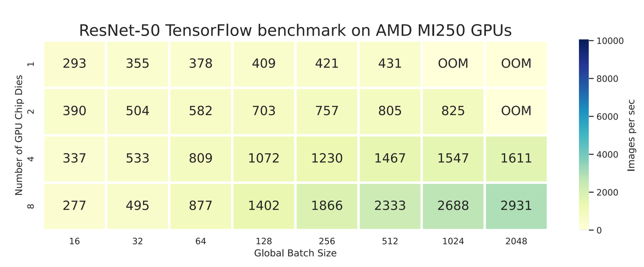
### NVIDIA A100 GPUs[5][8]



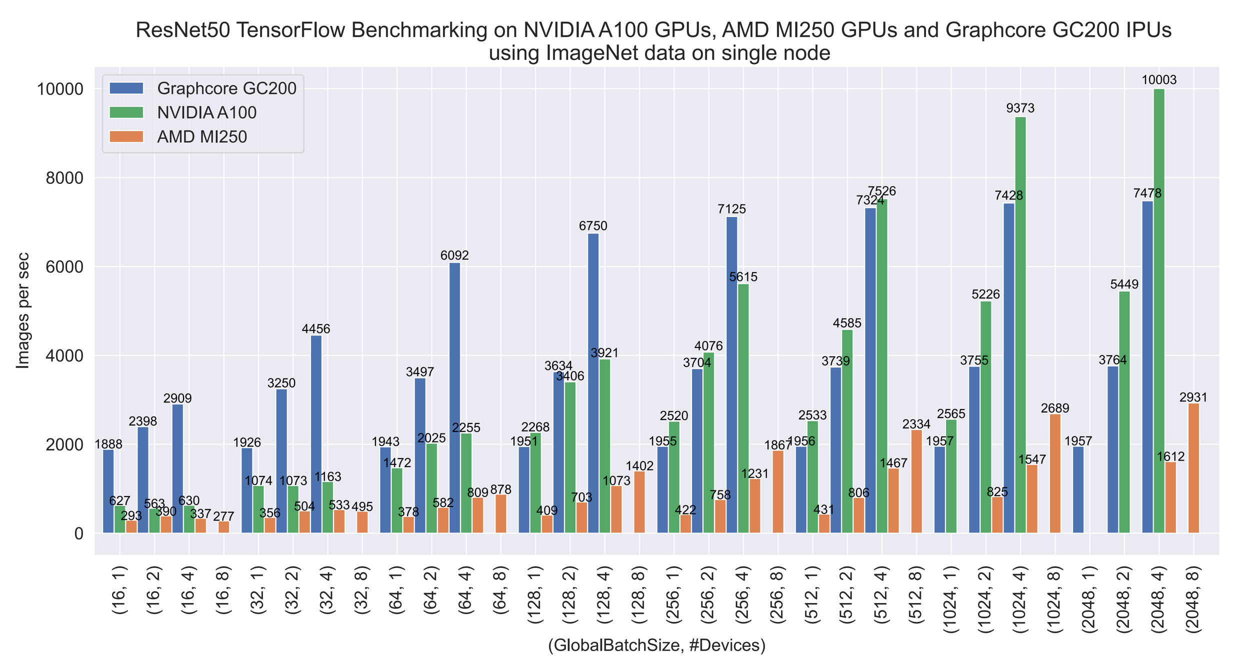

### Graphcore GC200 IPUs[6][7]




### AMD MI250 GPUs[5][7]




**Heatmaps: GlobalBatchSize vs. #Devices**
Throughput (images per sec) scales with global batch size and Data Parallelism for ResNet-50 model on single node; entire model fits on single device
⇒ #Devices = #Data Parallelism
Graphcore best for small batch sizes, NVIDIA for large; AMD significantly slower (reasons under evaluation)



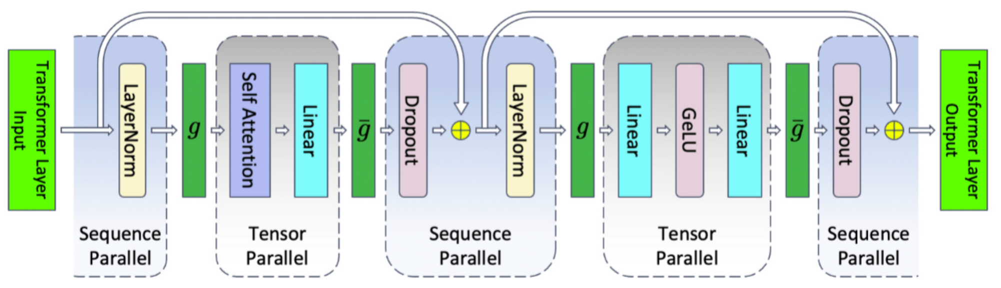

Bar graph comparing (GlobalBatchSize, #Devices) vs. Images per sec (throughput) for ResNet-50 model on NVIDIA[8], AMD[7] and Graphcore[7] devices on single nodes; entire model fits on single device
⇒ #Devices = #Data Parallelism

- Evaluation of new hardware architectures to test suitability for LLM
- Tests with simple TensorFlow ResNet-50 CNN benchmark using ImageNet data
- NVIDIA/AMD: Stock setup[5]
- Graphcore: Vendor/device-optimized setup[6]
- Using novel devices of JURECA DC Evaluation Platform[7] and JURECA DC[8]

*[5]: https://github.com/HelmholtzAI-FZJ/tf_cnn_benchmarks*
*[6]: https://github.com/graphcore/examples.git*
*[7]: JURECA Evaluation Platform: Additional hardware for benchmarking and testing at JSC*
*[8]: JURECA DC: Pre-Exascale Modular Supercomputer at JSC*

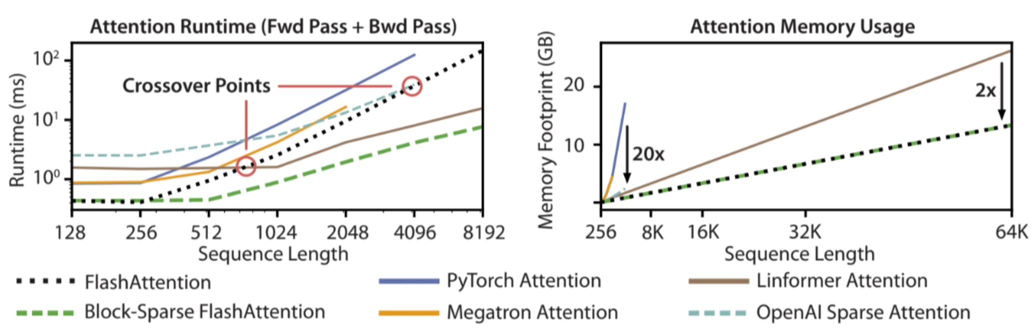
## Recent Advancements

**Sequence Parallelism:**
- Non-tensor parallel regions of transformer layer are **independent along sequence dimension**
- Prevent redundant storage of activations
- Selective re-computation of activation
- **5× memory reduction** with over 90% compute recovery from full activation re-computation.




Transformer layers with tensor and sequence parallelism.
*Reducing Activation Recomputation in Large Transformer Models;* https://arxiv.org/abs/2205.05198

**FlashAttention:**
- Attention algorithm with memory tiling between GPU high bandwidth memory (HBM) and GPU on-chip SRAM
- **20× memory efficient** and faster than standard attention without I/O optimisation
- Block-sparse flash attention is **faster than all implementations** across all sequence lengths




**Left**: Runtime of forward pass + backward pass. **Right:** Attention memory usage.
*FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness;* https://arxiv.org/abs/2205.14135

## Challenges

- Scarcity of evaluation tasks in languages other than English
- Availability of quality data
- Potential model biases
- Limited preprocessing filters for data
- Hardware robustness for large runs
- Energy consumptions: GPT-3 model training used approximately 936 MWh

## Next Steps

- Ablation studies on training objectives, optimizers and training parameters
- GPU communication and offloading using libraries (SHARP, UCC)
- CUDA Graphs
- High Performance Storage Tier NVMe cache[9]
- Implement **Recent Advancements**

*[9]: https://apps.fz-juelich.de/jsc/hps/juwels/cscratch.html#high-performance-storage-tier-cscratch*

## Acknowledgements

OpenGPT-X is funded by the Federal Ministry for Economic Affairs and Climate Action (BMWK) of Germany for the period 2022-2024. Compute time on the GCS Supercomputer JUWELS Booster at JSC is provided through the Gauss Centre for Supercomputing e.V.