TECHNISCHE UNIVERSITÄT WIEN

Exploring NCCL Tuning Strategies for Distributed Deep Learning

Majid Salimi Beni[1], Ruben Laso[2], Biagio Cosenza[3], Siegfried Benkner[2], and Sascha Hunold[1]

[1]Faculty of Informatics, TU Wien, Austria

[2]Faculty of Computer Science, University of Vienna, Austria

[3]Department of Computer Science, University of Salerno, Italy

The Fifteenth International Workshop on Accelerators and Hybrid Emerging Systems (AsHES)

Milan, June 3rd 2025

Today AI Models...

Large amount of **computing power** and **memory** needed!

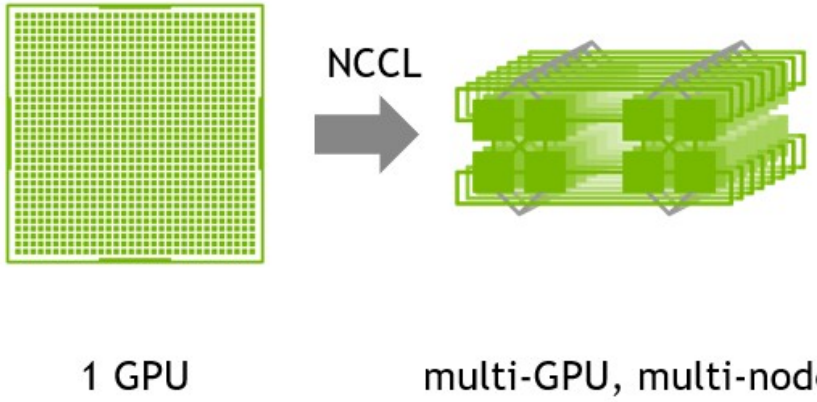
The Need for Distributed AI (Deep Learning)

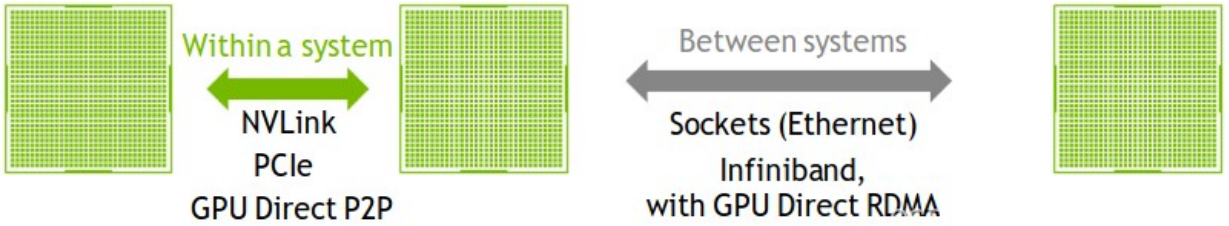
- It’s impossible to train recent AI models on a **single GPU/Node**
  - Computational power restrictions: Faster training time
  - Memory constraints: Large models
- HPC resources
- Distributed AI
  - New challenges: **Communication between GPUs**

https://leonardo-supercomputer.cineca.eu/hpc-system/

Distributed DL and Communication between GPUs

- **NCCL** (Nvidia Collective Communication Library)
  - The Central piece of software for distributed DL training
  - Handles **inter-GPU communication**
    - Can be inter and intra-node communication



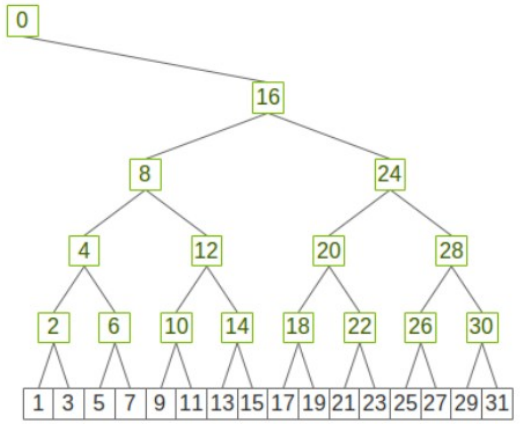


https://developer.nvidia.com/nccl

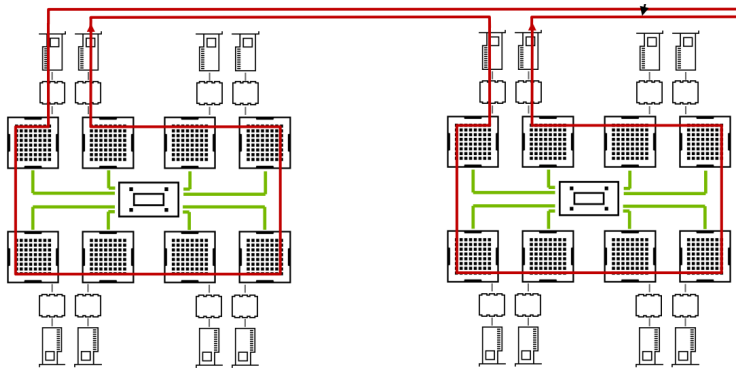
NCCL Parameters

- **Algorithm**: Ring, Tree,...
- **Network**: Infiniband, Ethernet
- **Protocol**: LL, Simple,...
- Network Interface selection
- NVLink vs Socket
- …
- **Around 90 parameters!**

Tree Algo



Ring Algo



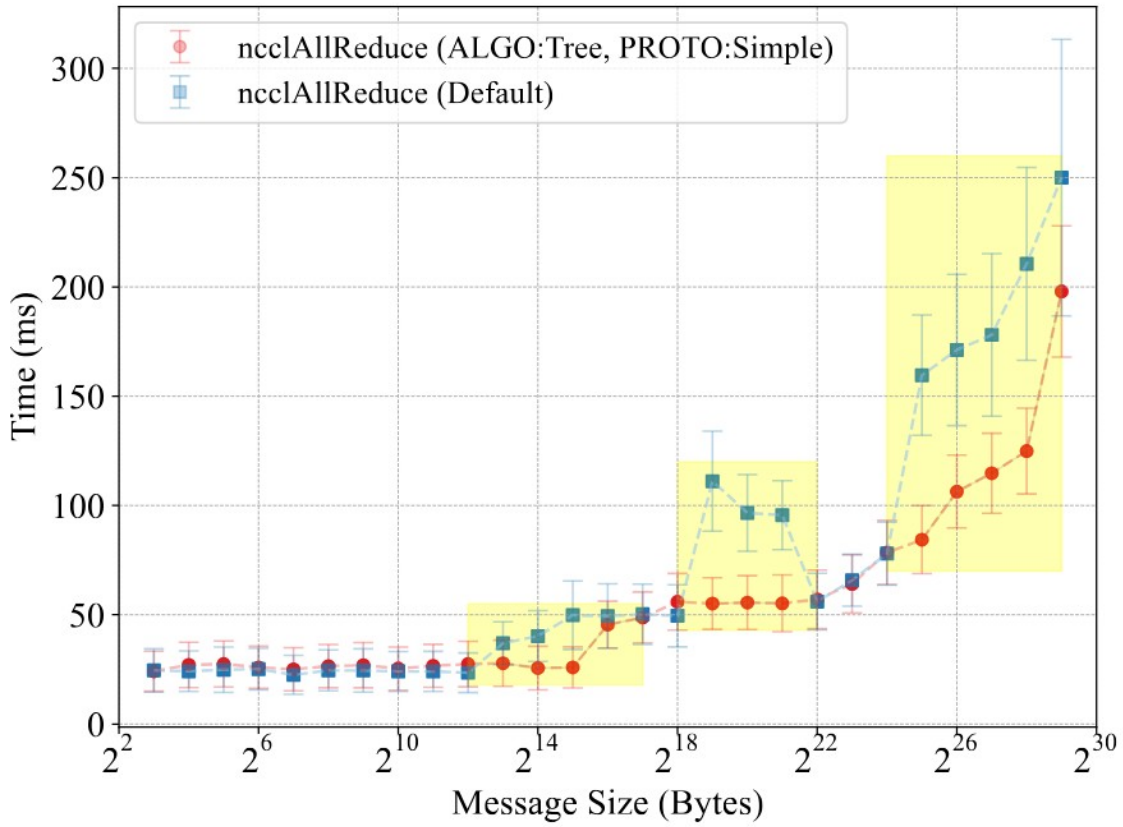
These configurations are **not well-tuned** for each application or compute cluster!

Tuning NCCL Parameters: The potential

- ncclAllreduce
  - 64 Nvidia A100 GPUs
  - Algorithm, Protocol: Tree,Simple

    vs
  - Default

- Default configuration is **not well-tuned** for algorithm and protocol for all message sizes

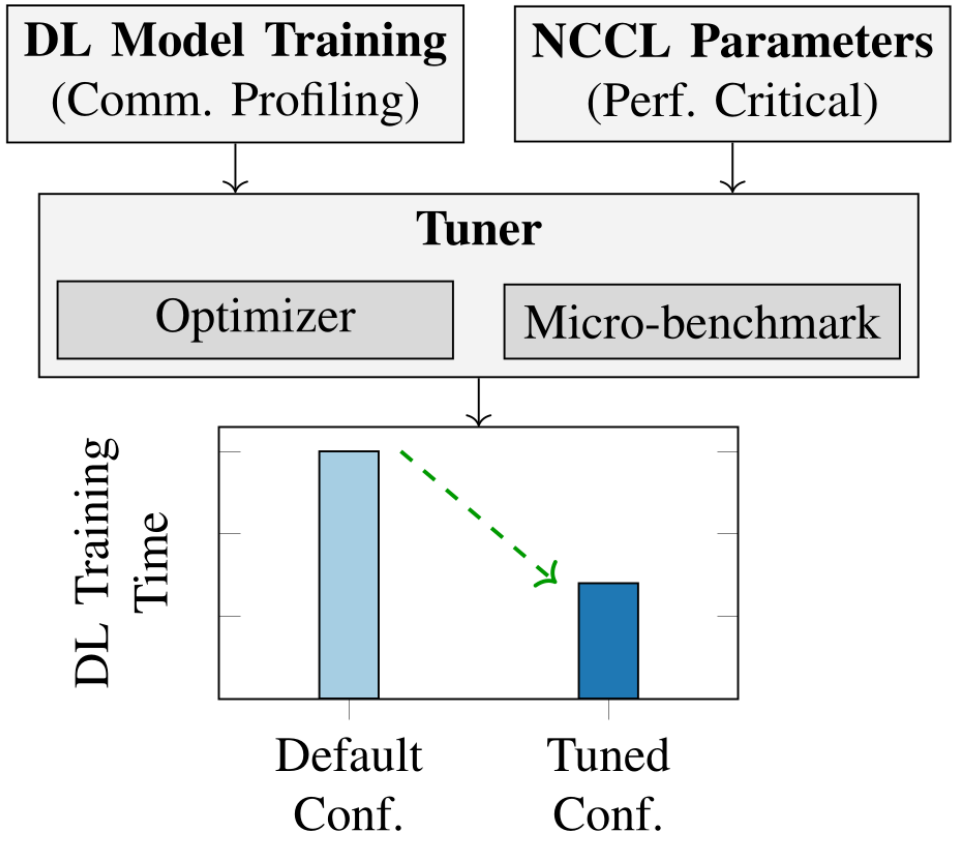


Results on 64 GPUs of Leonardo Supercomputer

Our Approach to Tuning NCCL

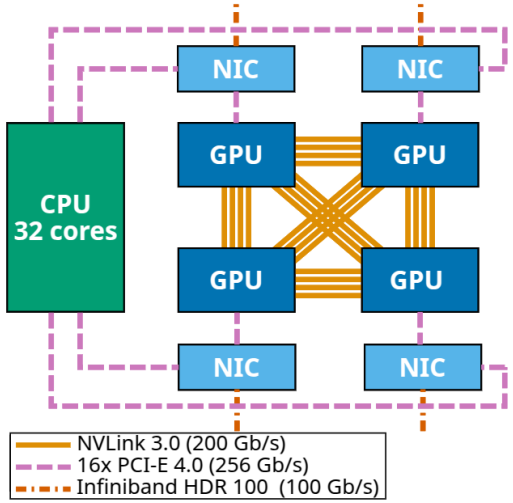
Tuning NCCL Parameters: Approach

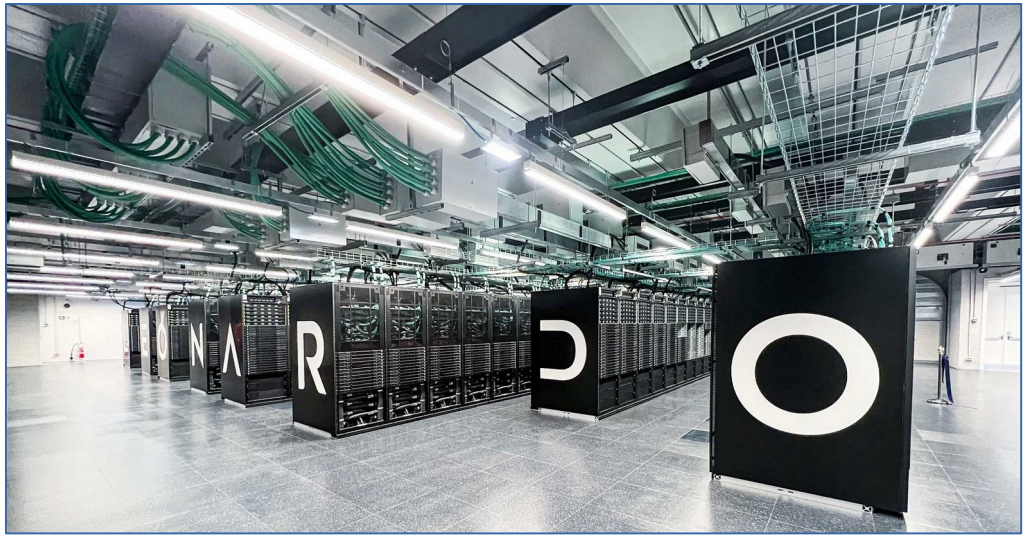
- **Profiling the communications** while training of deep learning models
- **Filtering** NCCL parameters
  - Excluding the irrelevant ones
  - From 90 to 45
- **Tuning**:
  - Offline tuning (Bayesian optimization)
  - NCCL Micro-benchmark
  - For each message size and collective, 30 minutes of tuning



Experimental Evaluation

- Results on 64 GPUs of Leonardo Supercomputer @CINECA
  - 16 nodes (4 GPUs per node)
- **Experiment 1**: Micro-benchmarking collectives: Tuned vs Default
- **Experiment 2:** Performance translation of tuned collectives to DL training



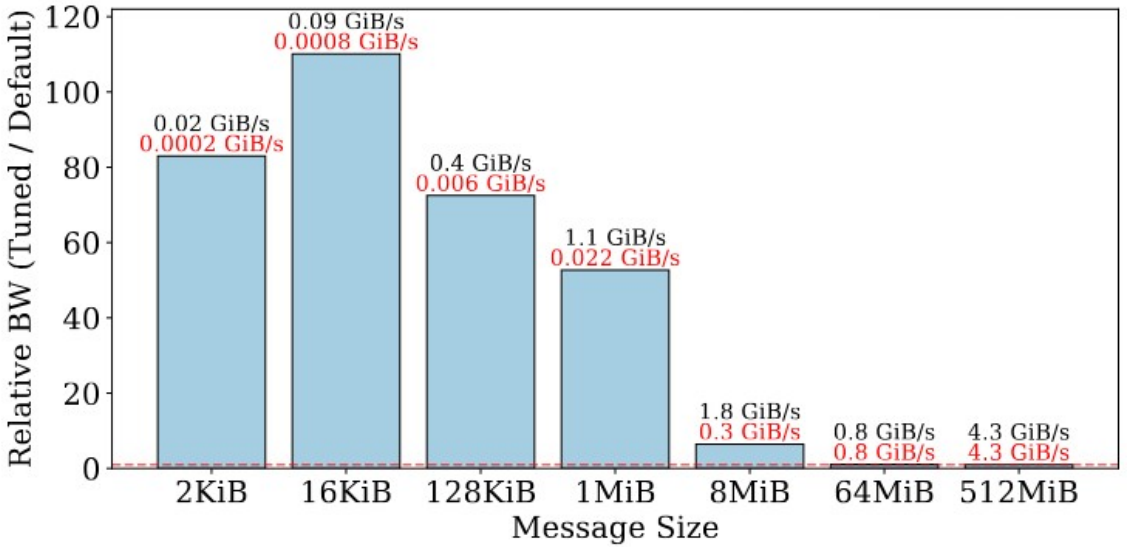


- https://www.nesite.com/en/technopole-bologna-in-the-heart-of-supercomputing-with-leonardo/
- De Sensi, Daniele, et al. "Exploring gpu-to-gpu communication: Insights into supercomputer interconnects." SC24.

Experiment 1: Tuning NCCL in Micro-benchmarks

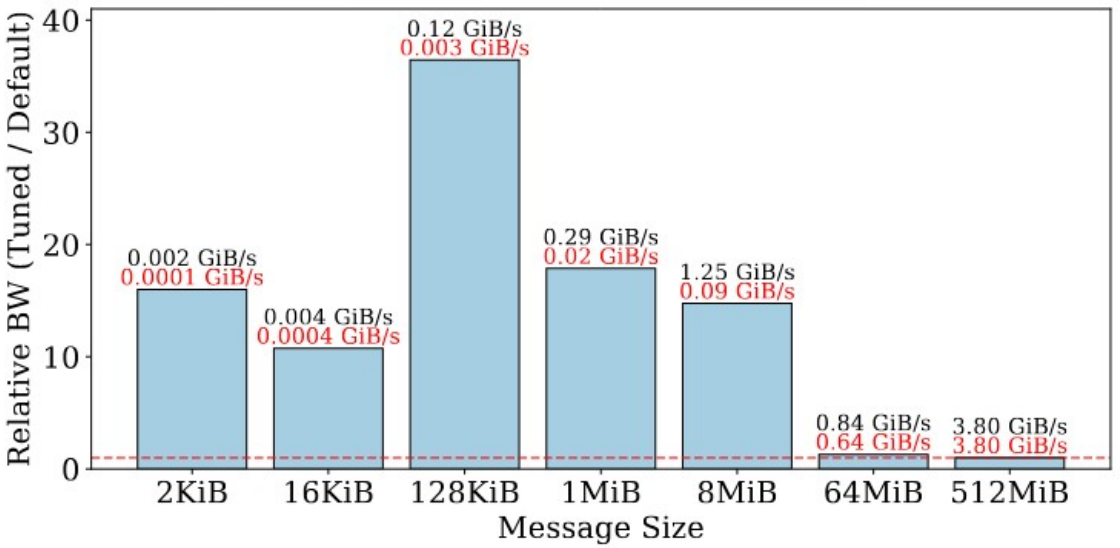
- Default vs Tuned (30 min for each message size)
- Results on 64 GPUs of Leonardo Supercomputer

At most:
112x improvement



(a) `ncclAllReduce`

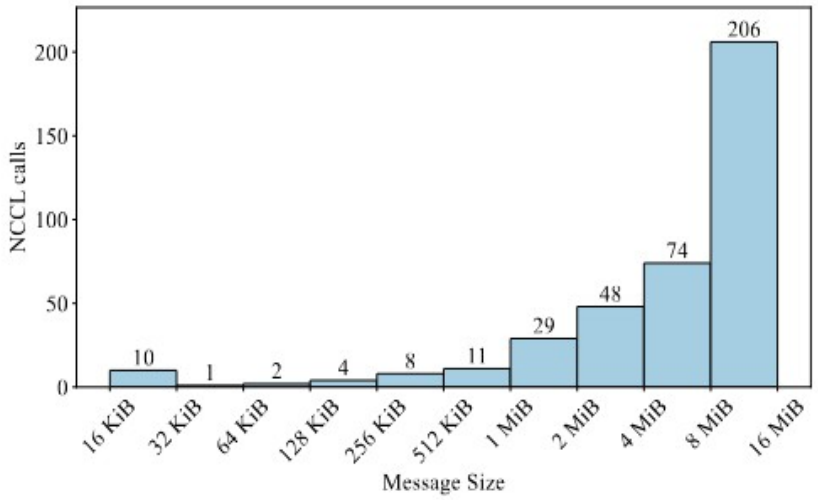
At most:
36x improvement



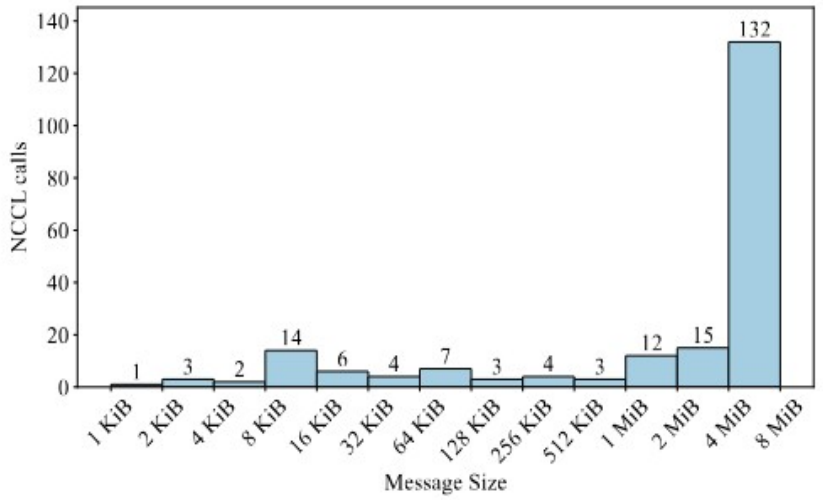
(b) `ncclAllGather`

Experiment 2: Profiling Communication in Distributed Deep Learning

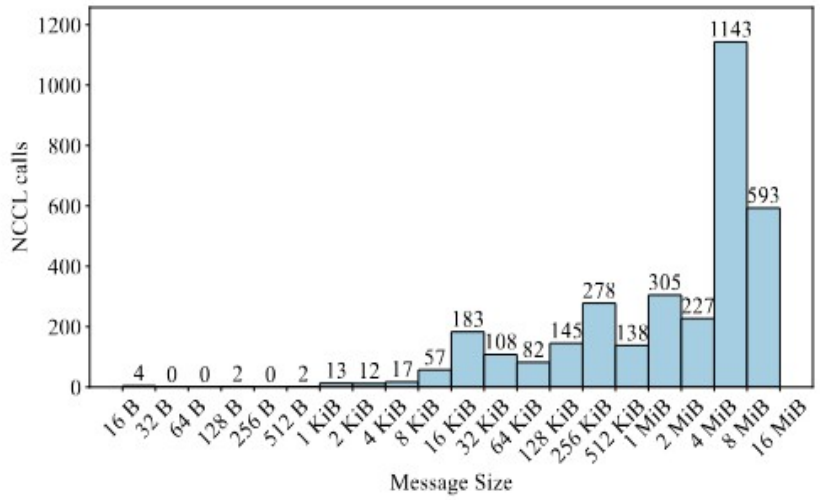
- **Step 1: Profile** to find dominant collective and message size
  - Tune NCCL for them



(a) DenseNet121



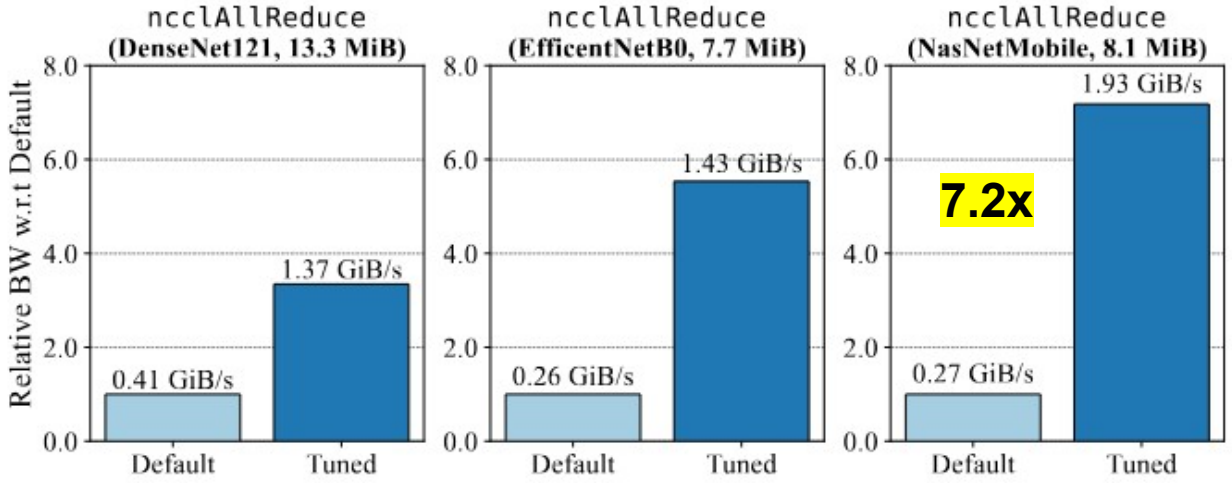
(b) EfficientNetB0



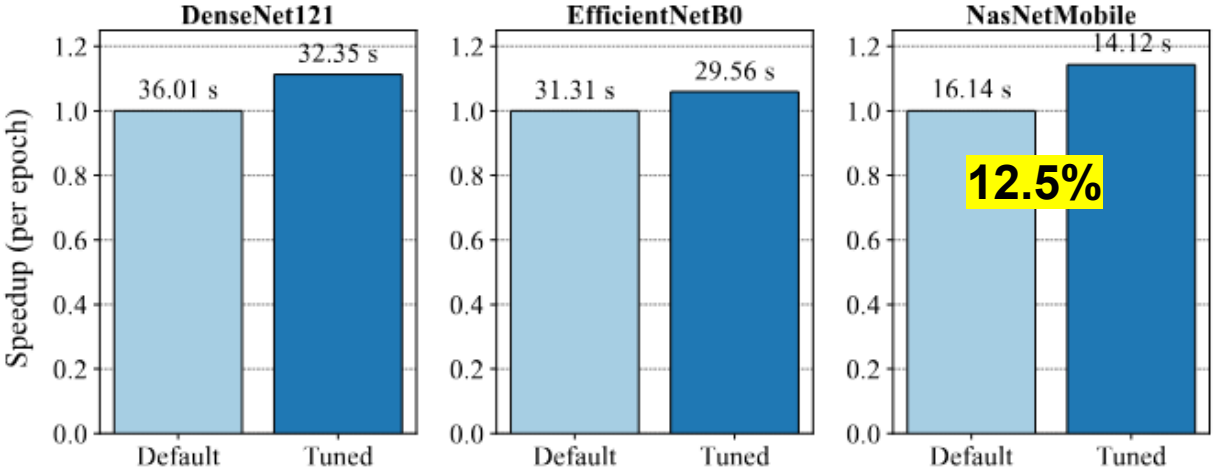
(c) NasNetMobile

Experiment 2: Tuned vs Default in DL Models

- **Step 2:** **Tune** for the dominant message size using micro-benchmarks
- **Step 3:** **Use tuned NCCL configurations** in the distributed DL training
- Even a small improvement in an epoch can highly impact long-running AI trainings



(a) Bandwidth in micro-benchmarks.



(b) Speedup per training epoch in deep learning models.

Summary and Conclusion

- Tuning potential of NCCL parameters
  - For different target systems
  - Different collective operations
- Tuning NCCL accelerates DL training
- Future work:
  - Statistical methods: Identify the most important parameters
  - Tuning NCCL for LLMs training

Thank you for your attention

Majid Salimi Beni, Ph.D.

Researh Group of Parallel Computing

Faculty of Informatics

TU Wien, Austria

Reach me at:

majid.salimibeni@tuwien.ac.at

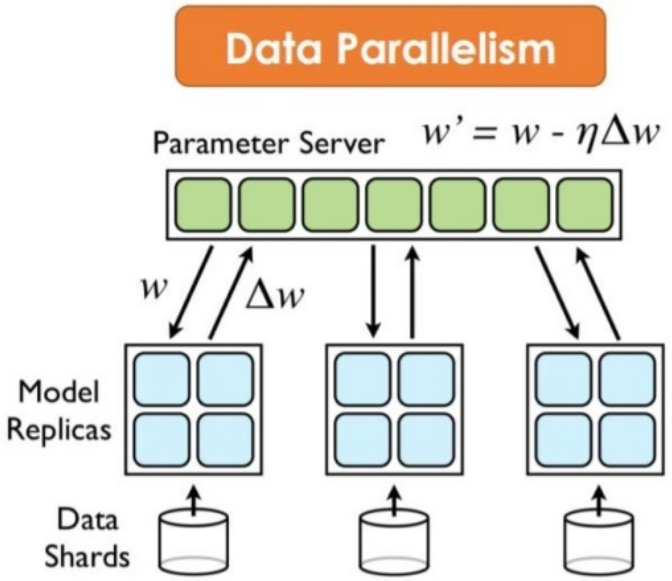
We thank **EuroHPC JU** for providing access to HPC resources!

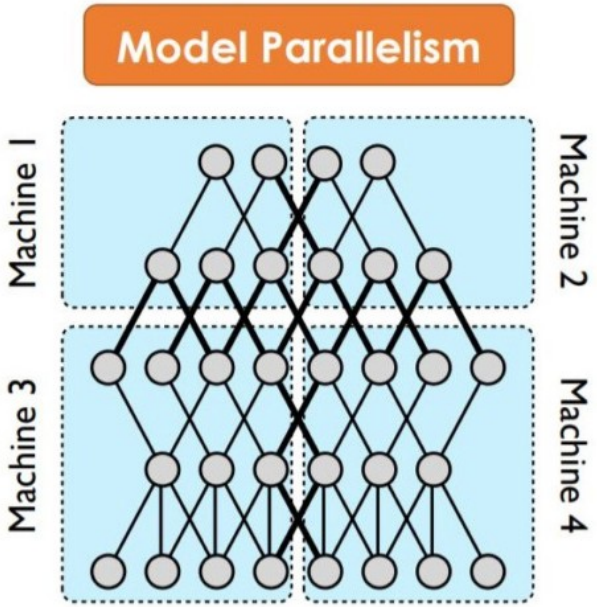
Backup 1 : Distributed AI (Deep Learning) Time

Metric	DeepSeek V3	Llama 3.1
Parameters	671B total (37B active per token)	405B
GPU Type	NVIDIA H800	NVIDIA H100
GPU Count	2,048	Up to 16,000
Training Duration	~2 months	~2.6 months (estimated)
Tokens Processed	14.8T	15.6T
GPU Hours	2.788M	~30.8M
Training Cost	~$5.6M	~$92.4M$–123.2M (estimated)

https://apxml.com/posts/training-cost-deepseek-v3-vs-llama-3

Backup 2: Distributed AI (Deep Learning)





Only suitable if model and mini-batches fit in the GPU memory

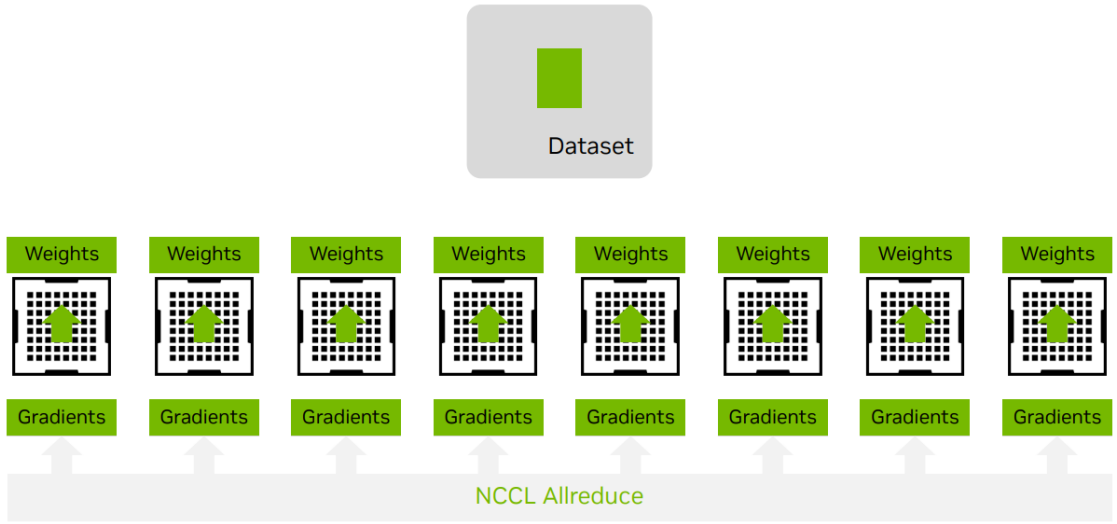
Reduced memory requirements → can train (very) big models

```
# Defining the base model:
model = torch.nn.Linear(20, 1)
# [..]
# Inside the distributed trainer
model_dist = torch.nn.parallel.DistributedDataParallel(model, device_ids=[rank])
```


Dean, Jeffrey, et al. "Large scale distributed deep networks." Advances in neural information processing systems 25 (2012).

Backup 3: Distributed DL and Communication between the GPUs

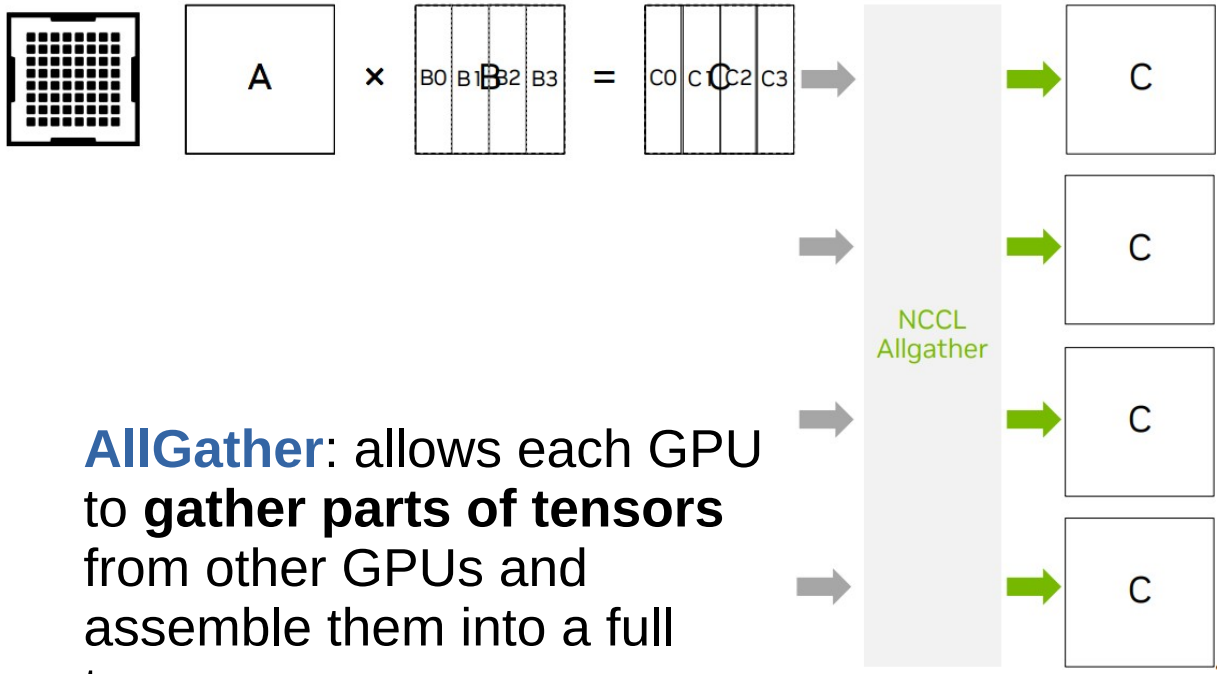
Data Parallel



AllReduce: **Synchronizing Gradients** in Data Parallel Training (summing them).

Each GPU receives the final averaged gradient and updates its model.

Model Parallel



AllGather: allows each GPU to **gather parts of tensors** from other GPUs and assemble them into a full tensor.

Broadcast – Distributing Model Weights to All GPUs

https://www.nvidia.com/en-us/on-demand/session/gtc24-s62129/