Lecture 4.1 - Distributed Deep Learning with High Performance Computing

End-to-End Machine Learning with High Performance and Cloud Computing - Tutorial IGARSS 2022

17.07.2022

Rocco Sedona
Jülich Supercomputing Centre - Forschungszentrum Jülich GmbH
University of Iceland

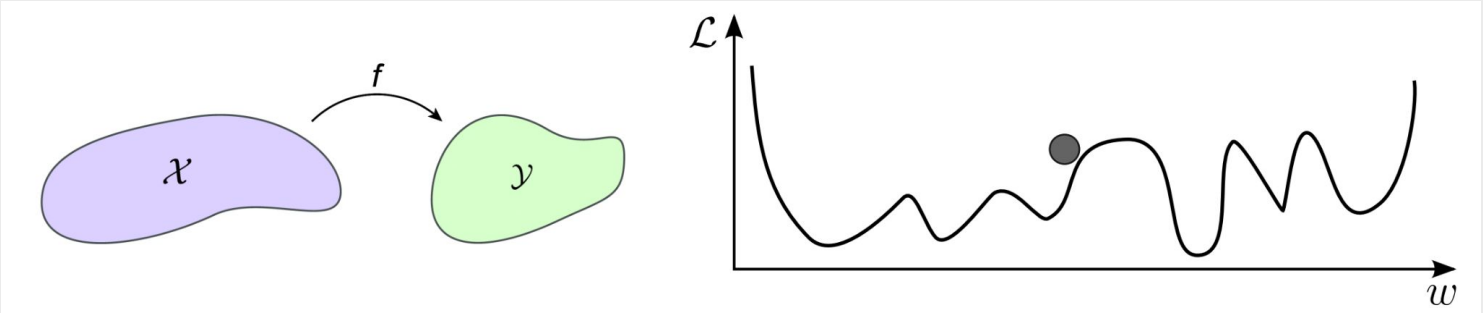
Outline

- Recap of basic concepts of Deep Learning
- Introduction to HPC
- MPI and other communication backends
- Introduction to Distributed Deep Learning
- Frameworks
- Final Remarks

Recap of basic DL concepts

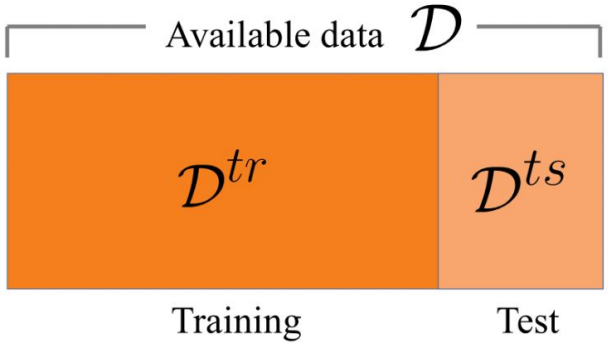
Optimization

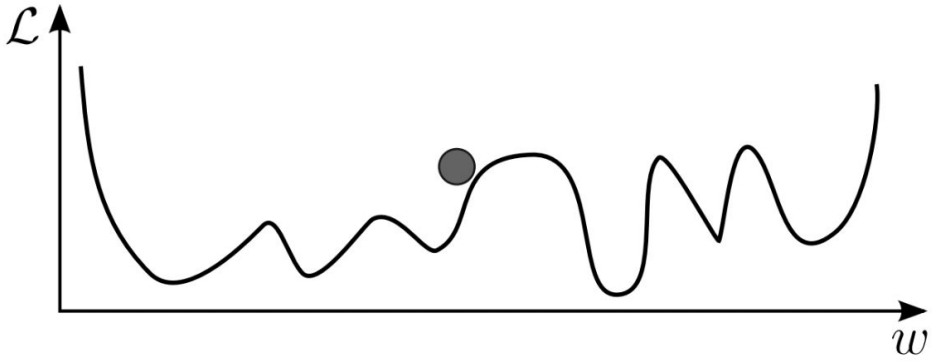
- Optimizing loss (objective) of a (complex) model f on data D
- a (complex) model: function (or distribution) family $f(X;\theta)$ $(p(X;\theta))$
- parameters θ are to adapt ("fit") given the data $X \in D$
- optimization: defining a loss $L(f(X;\theta), D)$
- loss L: measure of quality ("fit") of the model f in terms of a task solution on D
- Objective: minimize $L(f(X;\theta), D)$



Generalization

- Estimate $L(f(X;\theta), D^{unseen})$: aiming for good generalization capability
- General approach: split D into disjoint D_{tr} and D_{ts}, $D_{tr} \cap D_{ts} = \varnothing$
- train on D_{tr}
- generalization error on D_{ts} after training

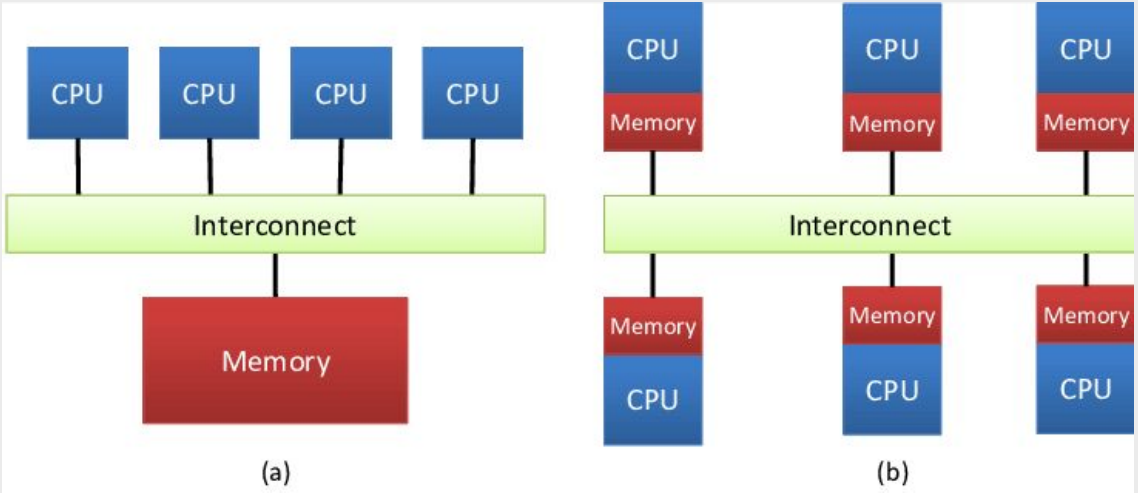




HPC

Hardware Levels of Parallelism

- (a) Single-machine (shared memory) (b) Multi-machine (distributed memory)



https://www.researchgate.net/publication/302245489_Adaptation_Strategies_in_Multiprocessors_System_on_Chip

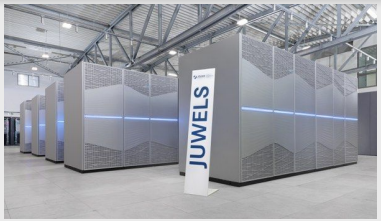
HPC

HPC at the frontline of computing power

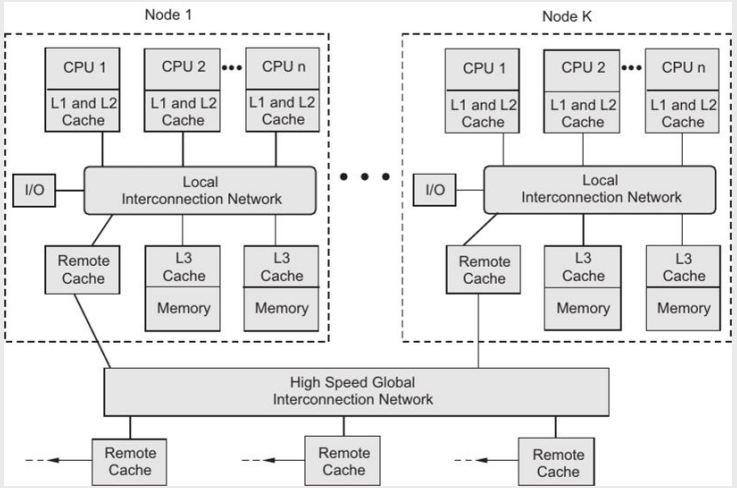
It includes work on ‘four basic building blocks’:

- Theory (numerical laws, physical models, speed-up performance, etc.)
- Technology (multi-core, supercomputers, networks, storages, etc.)
- Architecture (shared-memory, distributed-memory, interconnects, etc.)
- Software (libraries, schedulers, monitoring, applications, etc.)

Architecture: Shared-memory building blocks interconnected with a fast network (e.g., InfiniBand)



https://www.fz-juelich.de/de/ias/jsc

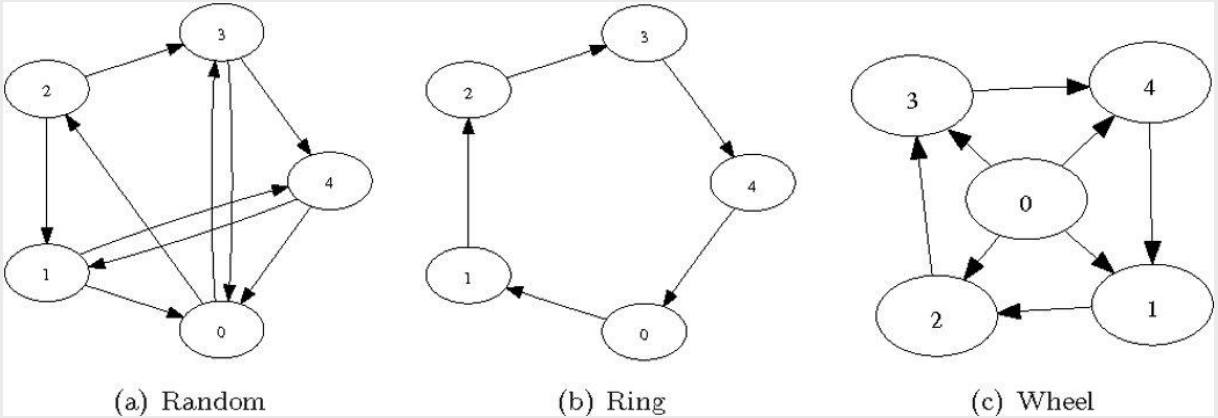


https://ebrary.net/206293/computer_science/distributed_shared_memory_multiprocessors_numa_model

Communication Backend

MPI

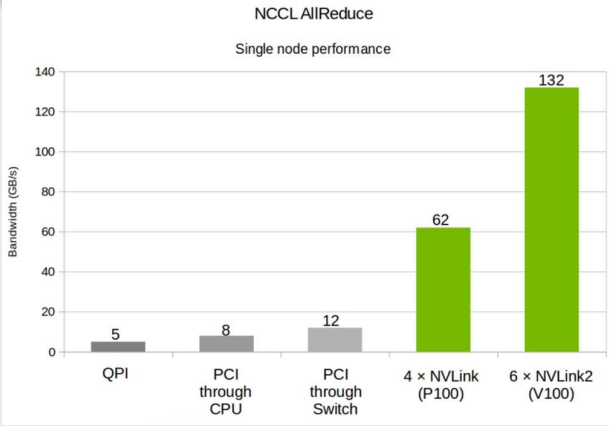
- MPI is a **standard** for **exchanging messages** between multiple computers running a parallel program across **distributed memory**
- Point-to-point and collective communication are supported
- **Different topologies** can be implemented
- Parallel I/O operations
- Blocking and non blocking statements

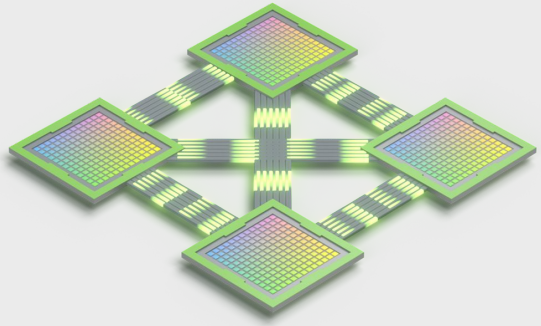


Hoefler, T., Rabenseifner, R., Ritzdorf, H., de Supinski, B. R., Thakur, R., & Träff, J. L. (2010). The scalable process topology interface of MPI 2.2. Concurrency and Computation: Practice and Experience, 23(4), 293–310. https://doi.org/10.1002/cpe.1643

NCCL

- NVIDIA Collective Communications Library (NCCL) [19]
- Provides **optimized implementation of inter-GPU communication** operations, such as allreduce and variants
- Optimized for **high bandwidth and low latency** over PCI and NVLink/NVSwitch high speed interconnect for intra-node communication (up to 16 GPUs)
- Sockets and InfiniBand for inter-node communication
- For a comparison between communication backends look at: [https://mlbench.github.io/2020/09/08/communication-backend-comparison/]





https://developer.nvidia.com/blog/scaling-deep-learning-training-nccl/

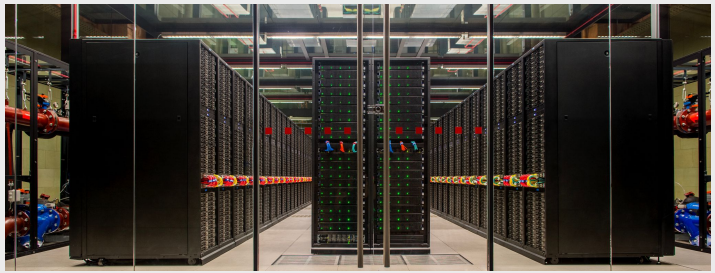
RCCL

- AMD's port of NCCL: ROCm Communication Collectives Library (RCCL) uses the same C API as NCCL
- NCCL APIs do not need to be converted

https://github.com/RadeonOpenCompute/ROCm

https://lumi-supercomputer.eu/easybuild-lumis-primary-software-installation-tool-introduced/

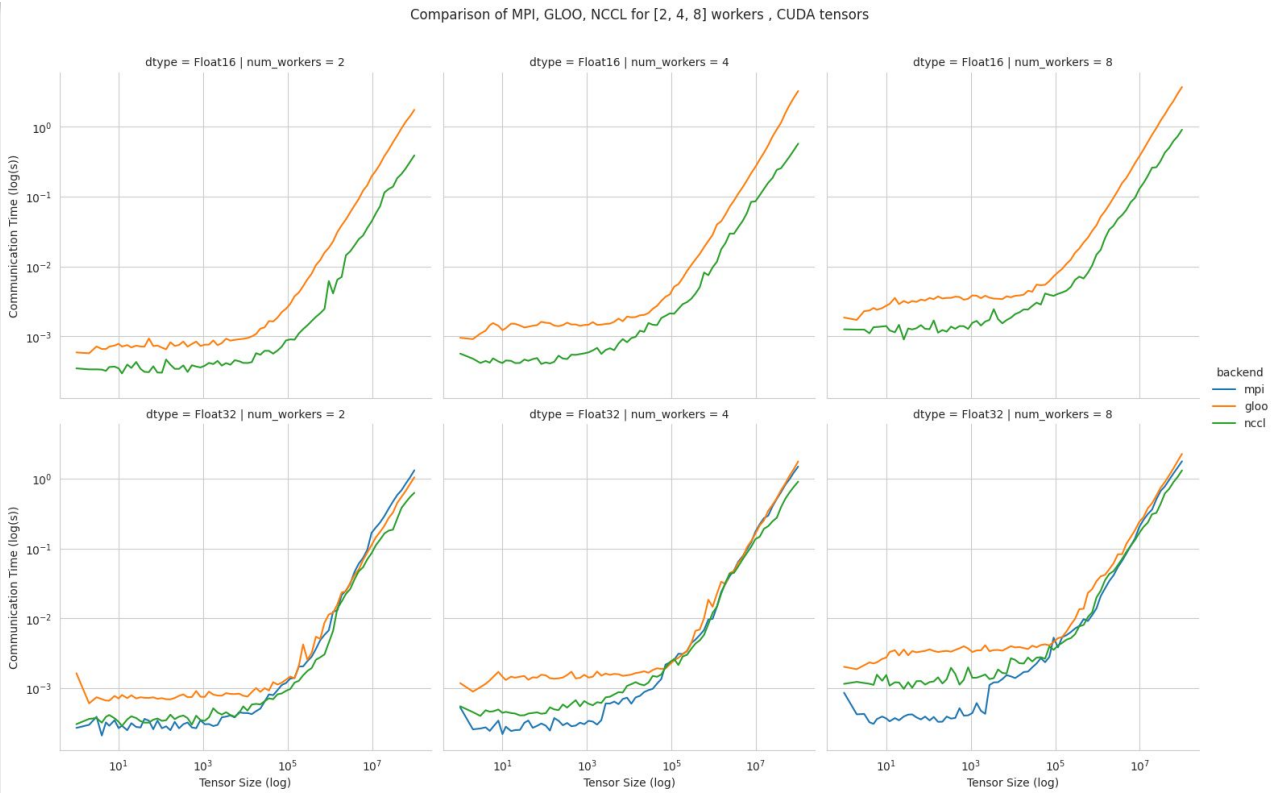
https://hwrig.com/amd-instinct-gpu-and-epyc-are-making-lumi-in-2021/



https://www.bsc.es/innovation-and-services/technical-information-cte-amd

Benchmark

- For a comparison between communication backends look at: [https://mlbench.github.io/2020/09/08/communication-backend-comparison/]
- MPI vs Gloo vs NCCL



Motivation

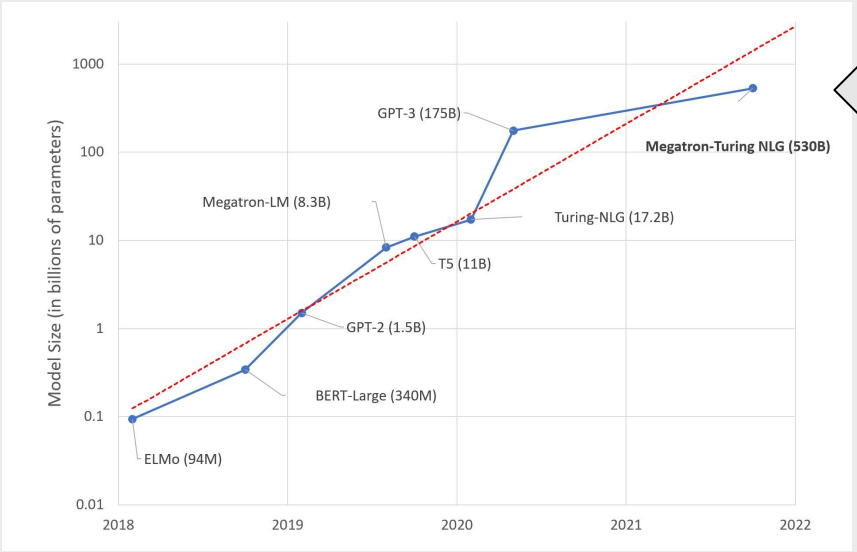
Motivation

In recent years almost exponential increase of number of parameters of the models

2020

https://www.microsoft.com/en-us/research/blog/a-deep-generative-model-trifecta-three-advances-that-work-towards-harnessing-large-scale-power/

2022

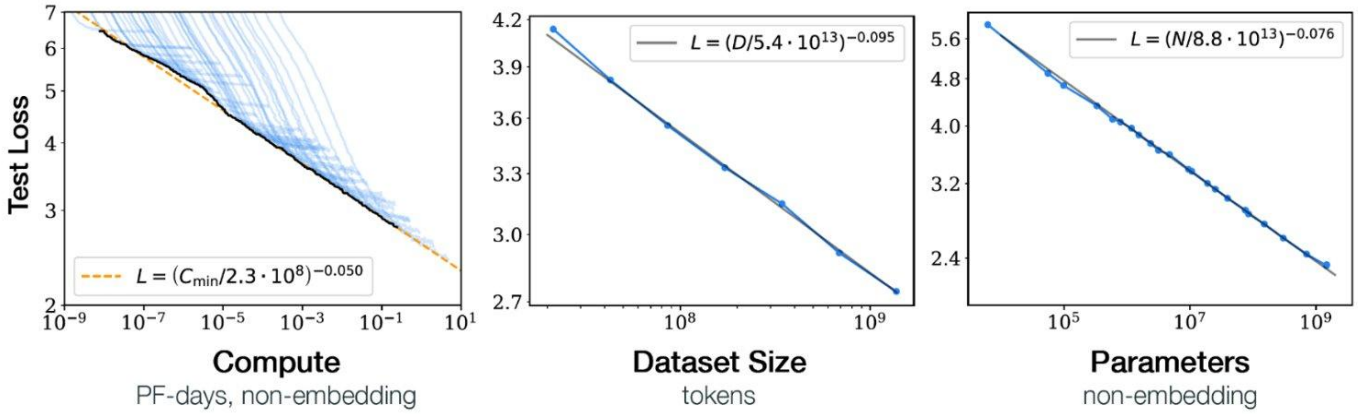


https://huggingface.co/blog/large-language-models

Motivation

- Bigger models require bigger datasets
- Consequence -> More resources are needed (both memory and computation power)

	Data Set	Type	Task	Size
small	MNIST	Image	Classification	55,000
small	Fashion MNIST	Image	Classification	55,000
small	CIFAR-10	Image	Classification	45,000
large	ImageNet	Image	Classification	1,281,167
large	Open Images	Image	Classification (multi-label)	4,526,492
large	LM1B	Text	Language modeling	30,301,028
large	Common Crawl	Text	Language modeling	~25.8 billion

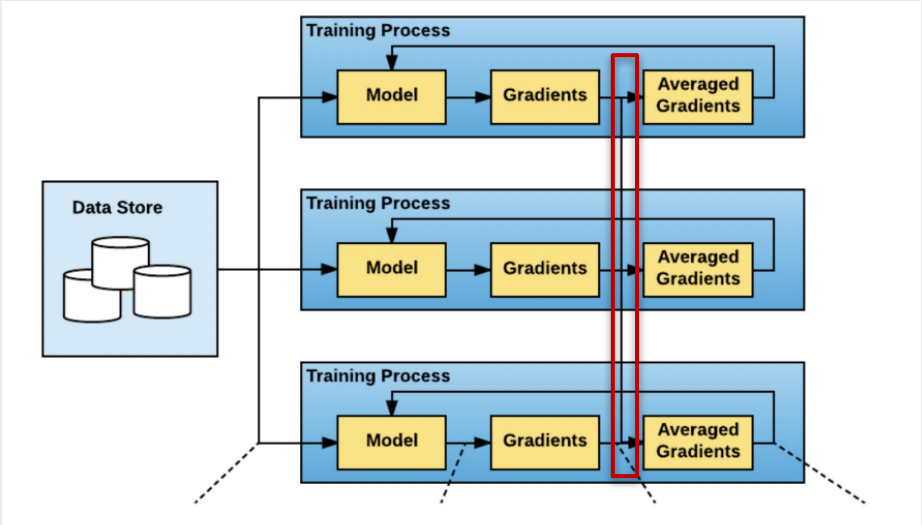


Kaplan et al., “Scaling Laws for Neural Language Models”, 2020, https://arxiv.org/abs/2001.08361

Distributed Deep Learning

Data Parallelism

- Concept: split the data
- The gradients for different batches of data are calculated separately on each node
- But averaged across nodes to apply consistent updates to the model copy in each node

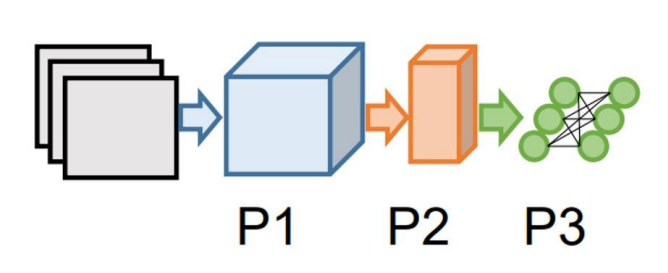


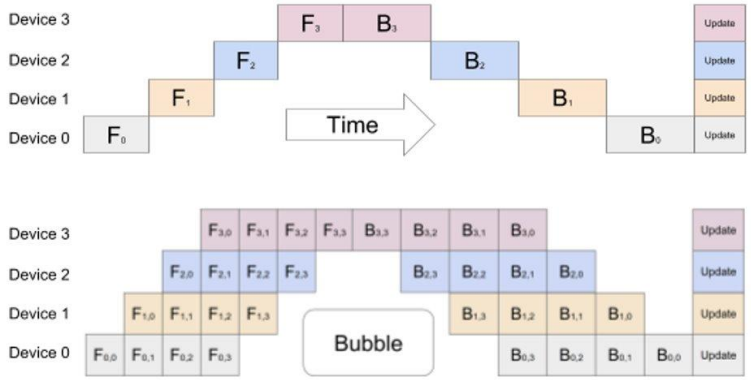
A. Sergeev and M. D. Balso, "Horovod: Fast and Easy Distributed Deep Learning in TensorFlow", arXiv:1802.05799, 2018

Model Parallelism

Concept: split the model
Pipelining:

- partitioning the DNN according to depth, assigning layers to specific processors
- overlapping computations, i.e., between one layer and the next (as data becomes ready)





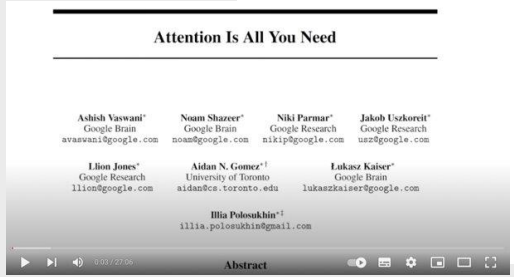
Top: The naive model parallelism strategy leads to severe underutilization due to the sequential nature of the network. Only one accelerator is active at a time. Bottom: GPipe divides the input mini-batch into smaller micro-batches, enabling different accelerators to work on separate micro-batches at the same time.

[https://huggingface.co/docs/transformers/parallelism]

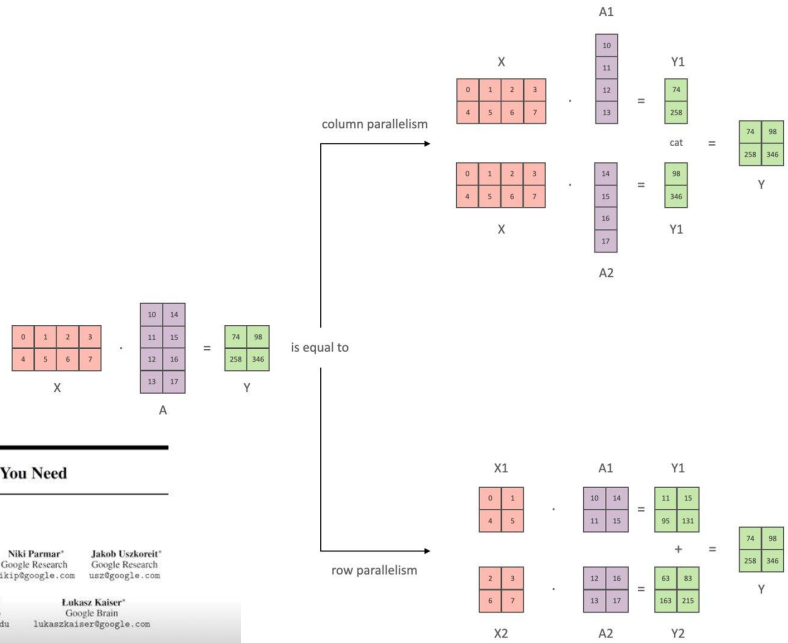
Model Parallelism

Tensor parallelism:

- matrix operations (f.e. matrix multiplication) can be split between multiple GPUs
- Scaling large transformers with multihead self-attention is based on this concept



[https://www.youtube.com/watch?v=iDulhoQ2pro&ab_channel=YannicKilcher]

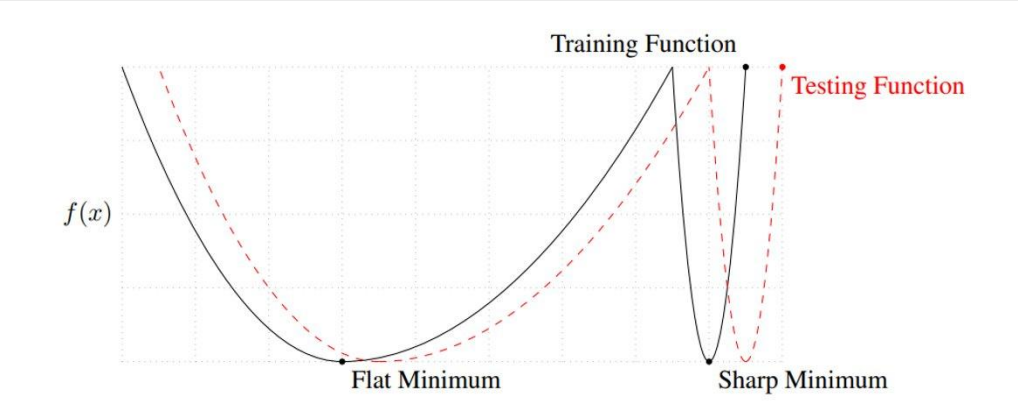


[https://huggingface.co/docs/transformers/parallelism]

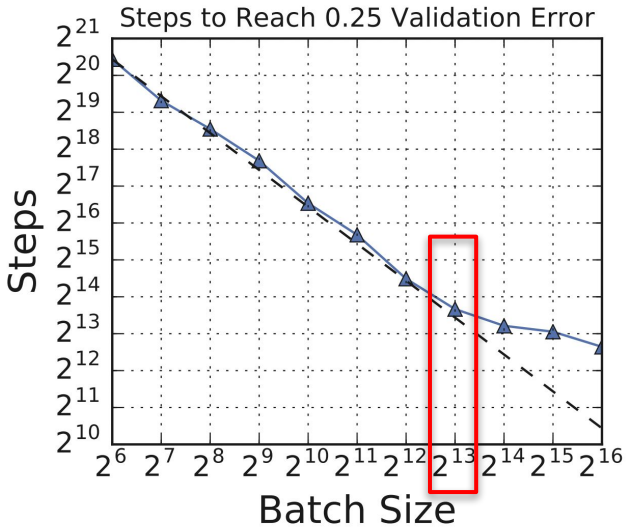
Challenges

- Poor generalization due to sharp minima

  [Hochreiter, Sepp and Schmidhuber, Jürgen. Flat minima. Neural Computation, 9(1):1–42, 1997]
- Time to accuracy does not decrease



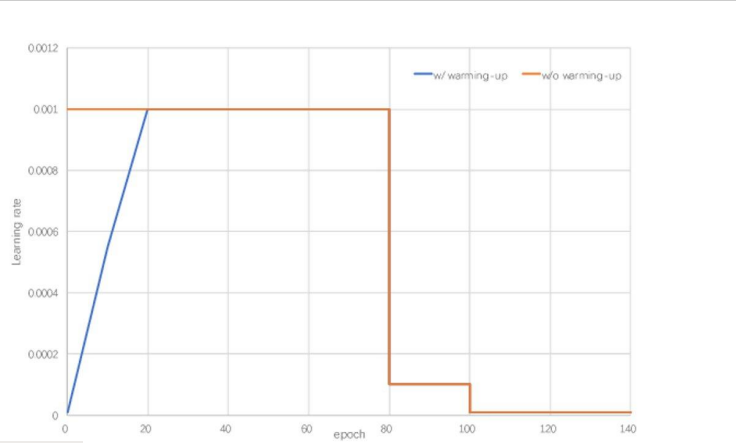
N. S. Keskar and D. Mudigere and J. Nocedal and M. Smelyanskiy and P.T.P. Tang, On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima, 2016



Shallue et al., 2019, https://arxiv.org/pdf/1811.03600.pdf

Solution

- For batch size < 8000
  - Scale learning rate
  - Warm-up
- For batch size > 8000
  - Choice of the optimizer:
    - LARS
    - LAMB
    - post-local SGD



Warm-up

	Hardware	Software	Batch size	Optimizer	# Steps	Time/step	Time	Accuracy
Goyal *et al.* [3]	Tesla P100 × 256	Caffe2	8,192	SGD	14,076	0.255 s	1 hr	**76.3** %
You *et al.* [11]	KNL × 2048	Intel Caffe	32,768	SGD	3,519	0.341 s	20 min	75.4 %
Akiba *et al.* [10]	Tesla P100 × 1024	Chainer	32,768	RMSprop/SGD	3,519	0.255 s	15 min	74.9 %
You *et al.* [11]	KNL × 2048	Intel Caffe	32,768	SGD	2,503	0.335 s	14 min	74.9 %
Jia *et al.* [12]	Tesla P40 × 2048	TensorFlow	65,536	SGD	1,800	0.220 s	6.6 min	75.8 %
Ying *et al.* [16]	TPU v3 × 1024	TensorFlow	32,768	SGD	3,519	**0.037 s**	2.2 min	**76.3** %
Mikami *et al.* [13]	Tesla V100 × 3456	NNL	55,296	SGD	2,086	0.057 s	2.0 min	75.3 %
Yamazaki *et al.* [14]	Tesla V100 × 2048	MXNet	81,920	SGD	1,440	0.050 s	**1.2 min**	75.1 %

Osawa et al., 2020

Is that all?

- Still ongoing research
- Well-establish optimizers can match new ones with enough *hyperparameter tuning*

Batch size	Step budget	LAMB	Adam
32k	15,625	91.48	**91.58**
65k/32k	8,599	90.58	**91.04**
65k	7,818	–	**90.46**

https://openreview.net/pdf?id=Kloou2uk_Rz

Frameworks

Frameworks

Horovod

- *Data parallel*, each GPU has a copy of the model and a chunk of the data
- Efficient *decentralized framework*,

based on MPI and NCCL libraries, where actors exchange parameters without the need of a parameter server

- Works on top of Keras, TensorFlow, PyTorch and Apache MXNet

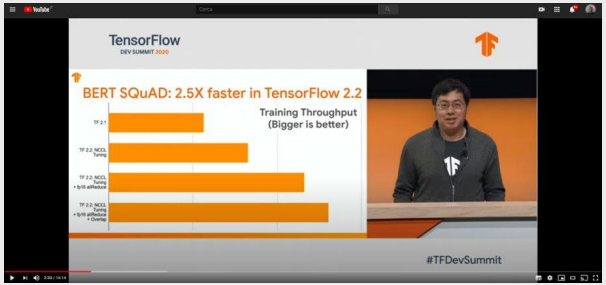
Tensorflow

- Parameter server for asynchronous training
- Mirrored strategy for synchronous training

Pytorch

- Distributed Data-Parallel Training (DDP)

A. Sergeev and M. D. Balso, "Horovod: Fast and Easy Distributed Deep Learning in TensorFlow", arXiv:1802.05799, 2018.



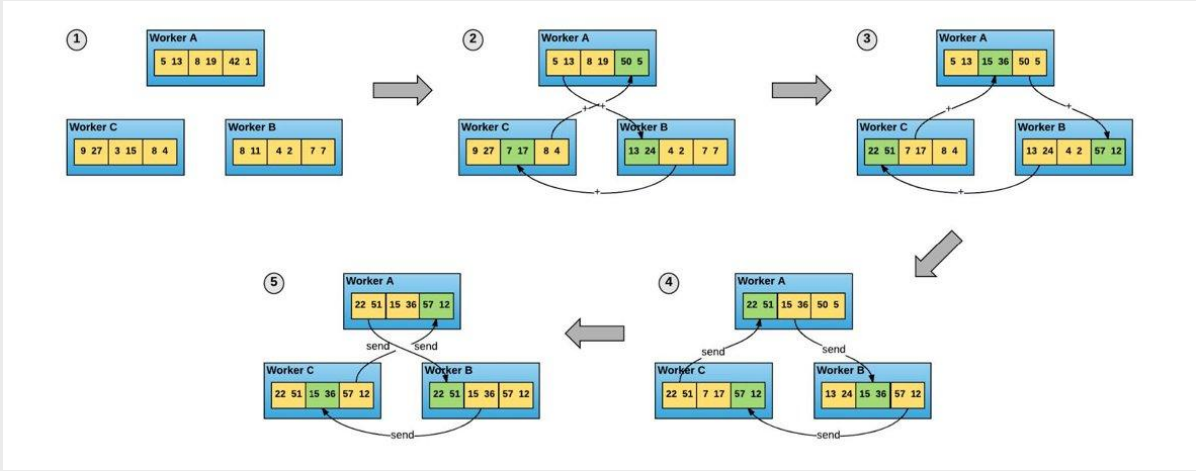
https://www.youtube.com/watch?v=6ovfZW8pepo&ab_channel=TensorFlow

Horovod

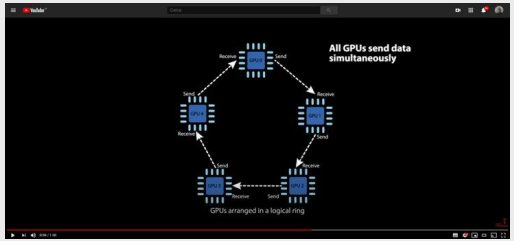
Ring allreduce

Two step process:

1. share-reduce step
2. share-only step



A. Sergeev and M. D. Balso, "Horovod: Fast and Easy Distributed Deep Learning in TensorFlow", arXiv:1802.05799, 2018



https://www.youtube.com/watch?v=4y0TDK3KoCA&t=585s&ab_channel=Uber Engineering

Distributed Training with TensorFlow

- tf.distribute.Strategy is a TensorFlow API that implements distributed training
- Easy to use and switching between strategies:
  - MirroredStrategy and MultiWorkerMirroredStrategy
  - TPUStrategy
  - ParameterServerStrategy
  - CentralStorageStrategy

- Use cluster resolver to read SLURM job configuration automatically

https://www.tensorflow.org/guide/distributed_training

Other Frameworks

TARANTELLA

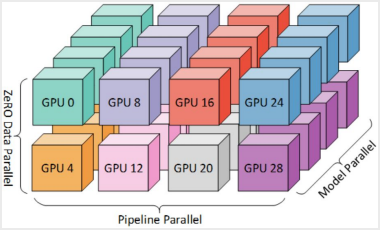
[https://github.com/cc-hpc-itwm/tarantella]

HEAT
Helmholtz Analytics Toolkit

https://github.com/helmholtz-analytics/heat

Colossal-AI

[https://github.com/hpcaitech/ColossalAI]



raysgd

https://docs.ray.io/

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

[https://github.com/NVIDIA/Megatron-LM]

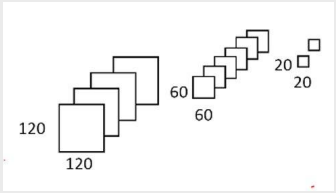
A Remote Sensing Use Case

BigEarthNet Classification

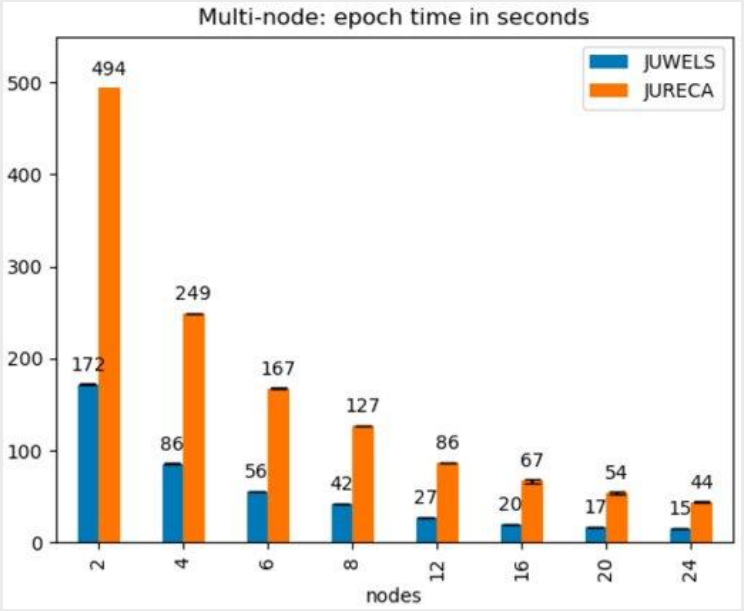
Dataset: BigEarthNet, Sentinel-2 Data Patches and Annotated with CORINE Land Covers

Model: ResNet50

0.74 F1-score up to 24 nodes - 96 GPUs with a global batch size of 8K samples



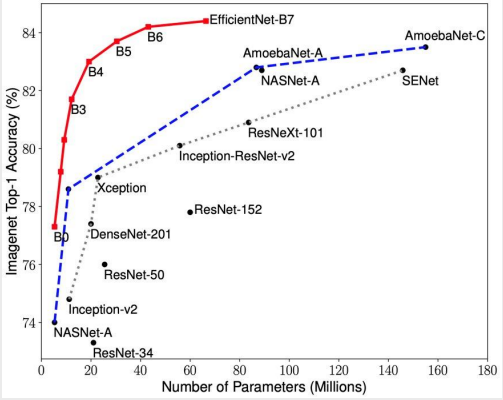
Patch and its dimension (px)



R. Sedona et al., Remote Sensing Big Data Classification with High Performance Distributed Deep Learning, 2019

Enhancements

- Adopted TensorFlow Dataset API to build a pipeline with integrated data augmentation, caching and prefetching of the data
- Deploying on 64 nodes / 256 GPUs of the Juwels Booster (Nvidia A100)
- New CNNs as EfficientNet, less parameters than ResNet, faster to train and higher accuracy
- Testing newer optimizers: LARS, LAMB, NovoGrad
- As the number of hyperparameters grows, there is the need to automatize the search for the optimal values (NAS)
- Hyper parameter tuning with **Ray Tune (embedded in Horovod)**: 'IGARSS2022 ACCELERATING HYPERPARAMETER TUNING OF A DEEP LEARNING MODEL FOR REMOTE SENSING IMAGE CLASSIFICATION', M. Aach, R. Sedona, A. Lintermann, G. Cavallaro, H. Neukirchen, M. Riedel, IGARSS2022 (accepted)



https://github.com/qubvel/efficientnet

Final Remarks

Final Remarks

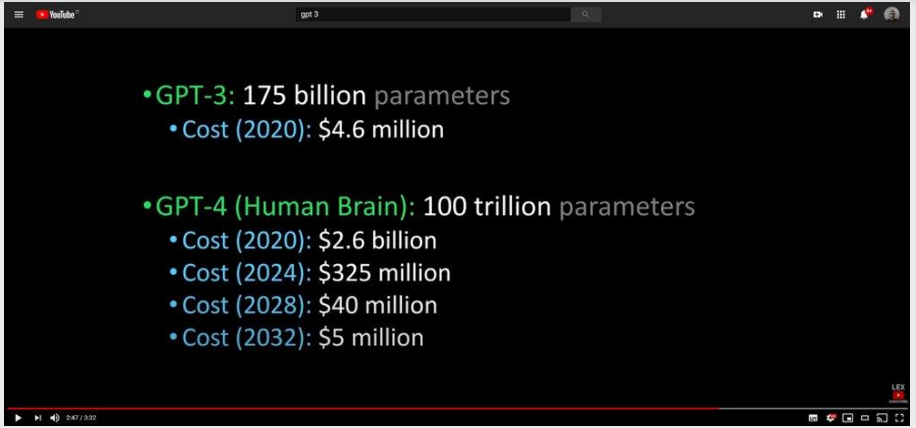
- The trend is to make distributed deep learning easier
- Not only frameworks, but integrated products
- Example: Dataflow-as-a-Service by SambaNova
- Intel's OpenAPI for heterogeneous computing
  [https://www.intel.com/content/www/us/en/developer/tools/oneapi/overview.html#gs.u1eb1g]
- AMD's GPUs using ROCm (similar to Nvidia's NCCL)

[f.e. https://www.bsc.es/innovation-and-services/technical-information-cte-amd]

[https://www.hpcwire.com/2020/12/09/ai-newcomer-sambanova-gas-product-lineup-and-offers-new-service/]

DL and Cloud Computing

- Trend towards cloud-based HPC
- What about costs?
- Let's have a look at **NCsv3-series** [25]
- **355 years** to train GPT-3 on a Tesla V100
- Training cost = 355Y×365D/Y×24H/D×0.9792$/H = **3.045.116$**



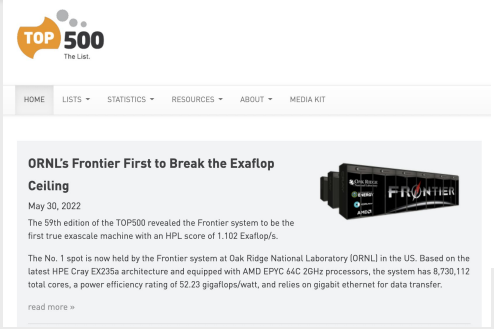
https://www.youtube.com/watch?v=kpiY_LemaTc&ab_channel=LexFridman

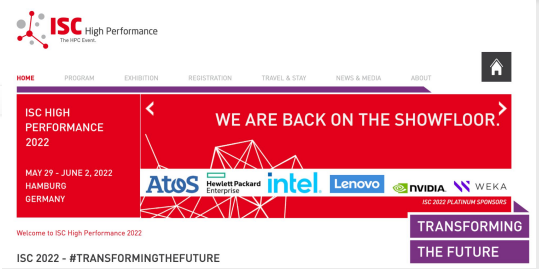
Add to estimate	Instance	Core	RAM	Temporary storage	GPU	Pay as you go	1 year reserved (% Savings)	3 year reserved (% Savings)	Spot (% Savings)
⊕	NC6s v3	6	112 GiB	736 GiB	1X V100	$3.06/hour	$1.9492/hour (~36%)	$0.9792/hour (~68%)	$0.306/hour (~90%)
⊕	NC12s v3	12	224 GiB	1.474 GiB	2X V100	$6.12/hour	$3.8984/hour (~36%)	$1.9585/hour (~68%)	$0.612/hour (~90%)
⊕	NC24rs v3	24	448 GiB	2.948 GiB	4X V100	$13.464/hour	$8.5766/hour (~36%)	$5.1002/hour (~62%)	$1.3464/hour (~90%)
⊕	NC24s v3	24	448 GiB	2.948 GiB	4X V100	$12.24/hour	$7.7970/hour (~36%)	$3.9169/hour (~68%)	$1.224/hour (~90%)

https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/

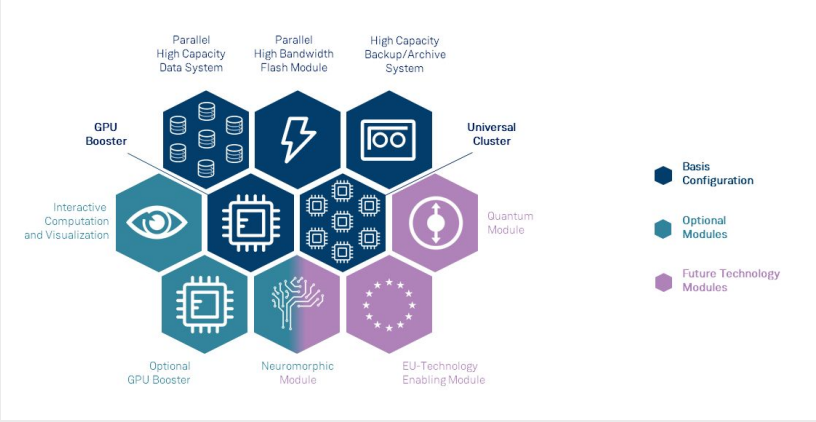
Towards Exascale

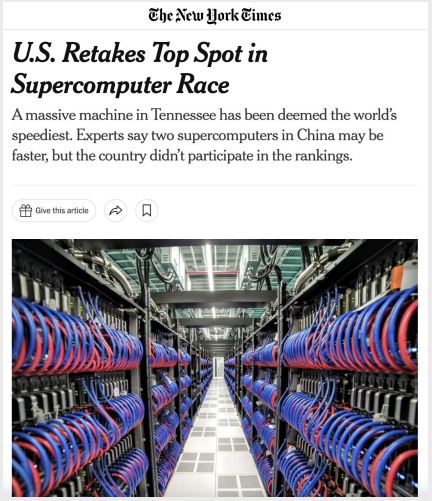
Frontier (First supercomputer to Break the Exaflop Ceiling at Oak Ridge National Laboratory (ORNL) in the US





Exascale Application Readiness
[https://www.olcf.ornl.gov/caar/frontier-caar/?fbclid=IwAR0JvTHz9rc_um_OGQbN28J8MDw5sv5yMF2OBWy2u5RKdMVyxODseWlnP7E]





https://www.fz-juelich.de/en/news/archive/press-release/2022/first-european-exascale-supercomputer-coming-to-julich

Final Remarks

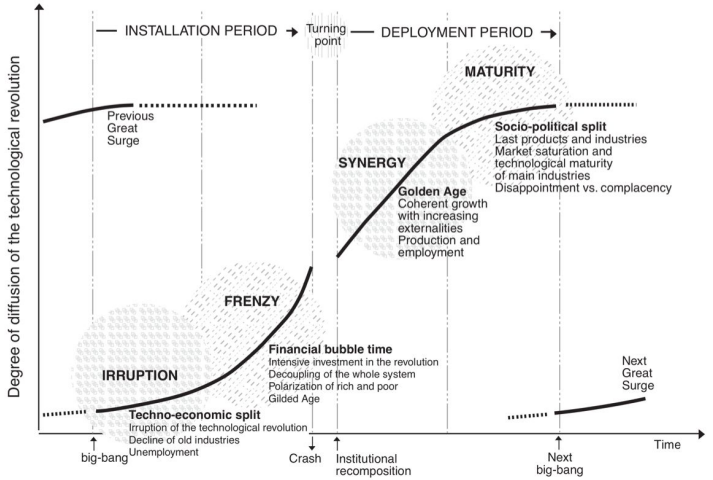
- Takeaways:
  - Frontier technology is fast paced
  - But successful solutions tend to become stable
  - Great opportunities for Distributed Deep Learning with the increased availability of computing resources

- Aknowledgement: Helmholtz AI Consultants

[https://www.helmholtz.ai/themenmenue/our-research/consultant-teams/helmholtz-ai-consultants-fzj/index.html]

[PRACE course "Introduction to Scalable Deep Learning" https://events.prace-ri.eu/event/1310/]

Figure 5.1 Recurring phases of each great surge in the core countries



Carlota Perez, 2002. "Technological Revolutions and Financial Capital," Books, Edward Elgar Publishing, number 2640.

drive. enable. innovate.

RAISE Center of Excellence

The CoE RAISE project have received funding from the European Union's Horizon 2020 – Research and Innovation Framework Programme H2020-INFRAEDI-2019-1 under grant agreement no. 951733

Follow us:

Practical

- STEP 1: git clone
- STEP 2: codice train.py (TF mirrored strategy)
- STEP 3: (if enought time) participants will create SLURM job script, otherwise we provide it
- STEP 4: run the code
- STEP 5: check error and output file and that the model is saved
- -> load the model onto AWS

Have scripts ready to show in case Jupyter JSC is down

https://www.tensorflow.org/guide/distributed_training#other_strategies