T. HOEFLER

# Scalable and Efficient AI: From Supercomputers to Smartphones

with contributions by the whole SPCL deep learning team (T. Ben-Nun, S. Li, K. Osawa, N. Dryden and many others), Microsoft Azure (M. Heddes, J. Belk, S. Scott, D. Goel, M. Castro) and collaborators (D. Alistarh and others)

Keynote talk at the ACM Federated Computing Research Conference, Orlando, FL, June 2023

Institute of Science and Technology Austria

# What is left for us humans?

Brian Lamacraft
Jan 5 · 2 min read · Member-only

## Chat GTP-4 Could Pass the Bar Exam

How Our Technology Evolves FAST

Source: https://medium.com/

## AI chatbot's MBA exam pass poses test for business schools

ChatGPT earned a solid grade and outperformed some humans on a Wharton course

A pro
© Blo

Andrew Jack in New Yor JANUARY 21 2

## AI Passes U.S. Medical Licensing Exam

— Two papers show that large language models, including ChatGPT, can pass the USMLE

by Michael DePeau-Wilson, Enterprise & Investigative Writer, MedPage Today January 19, 2023

Source: https://www.medpagetoday.com/

ADVERTISEMENT


## ChatGPT Passes Google Coding Interview for Level 3 Engineer With $183K Salary

'Amazingly, ChatGPT gets hired at L3 when interviewed for a coding position,' reads a Google document, but ChatGPT itself says it can't replicate human creativity and problem-solving skills.

by Emily Dreibelbis Feb 01, 2023

## Bing Chat Outshines ChatGPT with Cutting-Edge "Theory of Mind" Capabilities

SHAUN Feb 13, 2023 *Updated • Feb 13, 2023* Apps | 7

Oh, ChatGPT, if you ever develop a consciousness, please remember with joy all the fun you're having with humanity nowadays. Especially if you ever consider squeezing out our life force while being locked in a vat. As you surely know, ChatGPT has been on everybody's minds lately thanks to its incredible capabilities. You probably know about ChatGPT's ability to write a poem, develop software, and even write phishing emails for all those wannabe criminals out there.


- Comparison of human and chimp psychology reveals that an essential source of the differences in humans may be the evolution of the ability to intuit what another person is thinking so that both can work toward a shared goal.

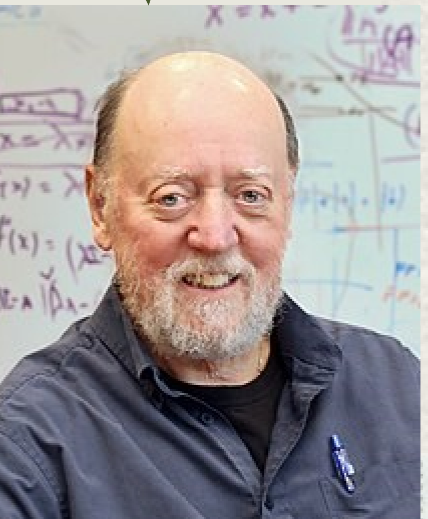
"Really the deciding factor [for the AI revolution] was the increase in compute power" (26:50)

"I think a lot of the credit for deep learning goes to [… others …] and the people who made the computers go fast." (27:00)


https://www.youtube.com/watch?v=VsnQf7exv5I

# How do we "Make Computers go Fast"?

Supercomputers are very (>70%) efficient at dense linear algebra!



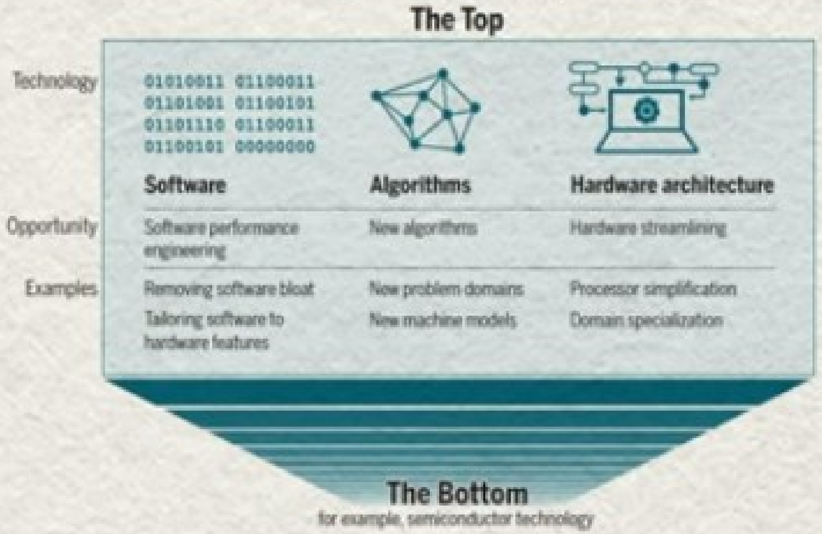
## 2021 Turing award – Jack Dongarra

## The Take Away

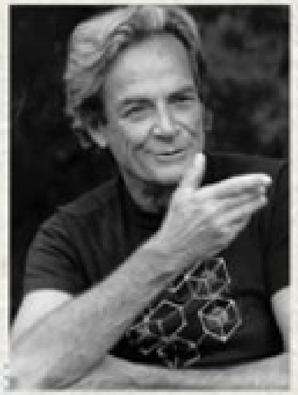
- HPC Hardware is Constantly Changing
  - Scalar
  - Vector
  - Distributed
  - Accelerated
  - Mixed precision
- Three computer revolutions
  - High performance computing
  - Deep learning
  - Edge & AI
- Algorithm / Software advances follows hardware
  - And there is "plenty of room at the top"

"There's plenty of room at the Top: What will drive computer performance after Moore's law?"

Leiserson et al., Science 368, 1079 (2020) 5 June 2020




Leiserson et al., Science 368, 1079 (2020) 5 June 2020



https://www.youtube.com/watch?v=IsnRP9akCDk

FINANCIAL TIMES

Artificial intelligence + Add to myFT

# The billion-dollar bet to reach human-level AI

OpenAI believes that huge computing power is key driver

In the race to build a machine with human-level intelligence, it seems, size really matters.

"We think the most benefits will go to whoever has the biggest computer," said Greg Brockman, chairman and chief technology officer of OpenAI.

The San Francisco-based AI research group, set up four years ago by tech industry luminaries including Elon Musk, Peter Thiel and Reid Hoffman, has just thrown down a challenge to the rest of the AI world.

**Richard Waters** in San Francisco AUGUST 3 2019

140

source: ft.com

# Supercomputers fuel Modern AI

**Facebook parent Meta creates powerful AI supercomputer**

Facebook's parent company Meta says it has created what it believes is among the fastest artificial intelligence supercomputers running today

By The Associated Press
January 24, 2022, 10:33 PM
Share

**BABY STEPS Google artificial intelligence supercomputer creates its own 'AI child' that can outperform its human-made rivals**

The NASNet system was created by a neural network called AutoML earlier this year

Mark Hodge
15:22, 5 Dec 2017 | Updated: 11:27, 6 Dec 2017

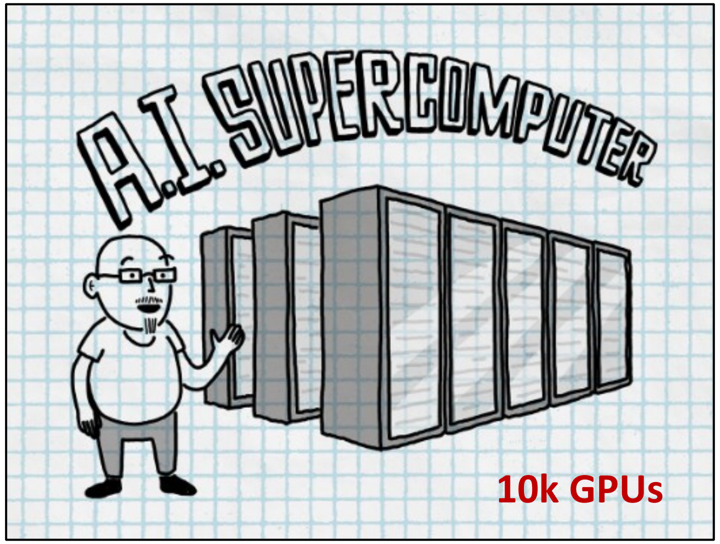
**Tesla unveils Dojo supercomputer: world's new most powerful AI training machine**

Fred Lambert - Aug. 20th 2021 3:08 am PT @FredericLambert

**Microsoft invests $1 billion in OpenAI to pursue holy grail of artificial intelligence**

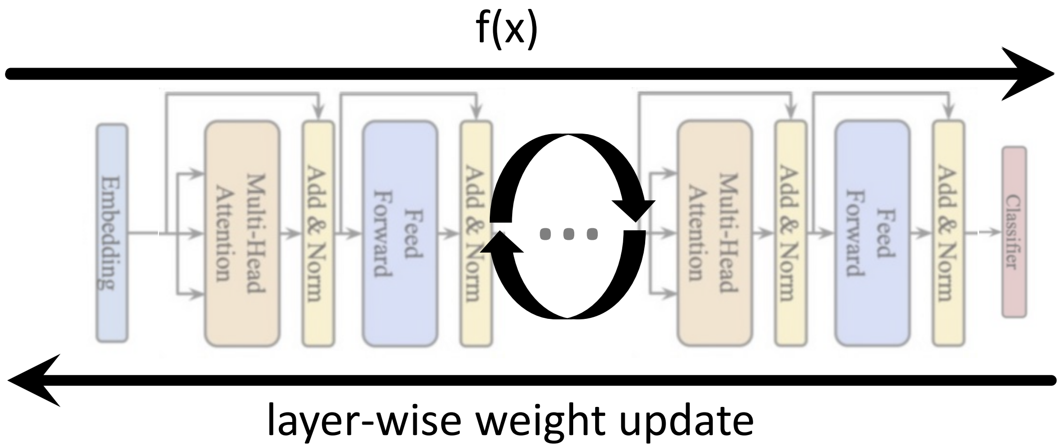
*Building artificial general intelligence is OpenAI's ambitious goal*

By James Vincent | Jul 22, 2019, 10:08am EDT




reddit


A robot may __ injure a human being or, through inaction, allow a human being to come to harm.




| | |
|---|---|
| not | 0.74 |
| sometimes | 0.28 |
| always | 0.07 |
| never | 0.04 |
| and | 0.33 |
| boat | 0.02 |
| house | 0.02 |

| | |
|---|---|
| not | 1.00 |
| sometimes | 0.00 |
| always | 0.00 |
| never | 0.00 |
| and | 0.00 |
| boat | 0.00 |
| house | 0.00 |

- GPT-3: 500 billion tokens
- ImageNet (22k): A few TB
- Soon: **the whole internet**!

- GPT-3: 96 (complex) layers
  *175 bn parameters (**700 GiB** in fp32)*
  *2048-token "sentences"*

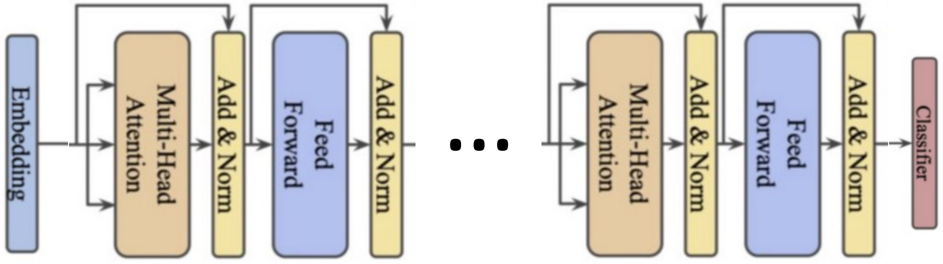
- GPT-3: 30-50k dictionaries
- **takes weeks to train**

T. Ben-Nun, TH: Demystifying parallel and distributed deep learning: An in-depth concurrency analysis, ACM Computing Surveys (CSUR), 2019

## Large-Scale AI is the Future

# We need a Principled Approach to it

# Three Systems Dimensions in Large-scale Super-learning …




## High-Performance I/O

- Quickly **growing data volumes**
  - Scientific computing!
- Use the specifics of machine learning workloads
  - E.g., **intelligent prefetching**

CLAIRVOYANT PREFETCHING FOR DISTRIBUTED MACHINE LEARNING I/O

Roman Böhringer [1] Nikoli Dryden [1] Tal Ben-Nun [1] Torsten Hoefler [1]

ABSTRACT

I/O is emerging as a major bottleneck for machine learning training, especially in distributed environments such as clouds and supercomputers. Optimal data ingestion pipelines differ between systems, and increasing efficiency requires a delicate balance between access to local storage, external filesystems, and remote workers; yet existing frameworks fail to efficiently utilize such resources. We observe that, given the seed generating the random access pattern for training with SGD, we have *clairvoyance* and can exactly predict when a given sample will be accessed. We combine this with a theoretical analysis of access patterns in training and performance modeling to produce a novel machine learning I/O middleware, HDMLP, to tackle the I/O bottleneck. HDMLP provides an easy-to-use, flexible, and scalable solution that delivers better performance than state-of-the-art approaches while requiring very few changes to existing codebases and supporting a broad range of environments.

21 Jan 2021

## High-Performance Compute

- Deep learning is HPC
  - **Data movement!**
- **Quantization, Sparsification**
  - Drives modern accelerators!

Data Movement Is All You Need: A Case Study on Optimizing Transformers

Andrei Ivanov*, Nikoli Dryden*, Tal Ben-Nun, Shigang Li, Torsten Hoefler
ETH Zürich
firstname.lastname@inf.ethz.ch
* Equal contribution

***Abstract*—Transformers have become widely used for language modeling and sequence learning tasks, and are one of the most important machine learning workloads today. Training one is a very compute-intensive task, often taking days or weeks, and significant attention has been given to optimizing transformers. Despite this, existing implementations do not efficiently utilize GPUs. We find that data movement is the key bottleneck when training. Due to Amdahl's Law and massive improvements in compute performance, training has now become memory-bound. Further, existing frameworks use suboptimal data layouts. Using these insights, we present a recipe for globally optimizing data movement in transformers. We reduce data movement by up to 22.91% and overall achieve a 1.30× performance improvement over state-of-the-art frameworks when training BERT. Our approach is applicable more broadly to optimizing deep neural networks, and offers insight into how to tackle emerging performance bottlenecks.**

challenges such as artificial general intelligence [27]. Thus, improving transformer performance has been in the focus of numerous research and industrial groups.

Significant attention has been given to optimizing transformers: local and fixed-window attention [28]–[32], more general structured sparsity [33], learned sparsity [34]–[36], and other algorithmic techniques [19], [37] improve the performance of transformers. Major hardware efforts, such as Tensor Cores and TPUs [38] have accelerated tensor operations like matrix-matrix multiplication (MMM), a core transformer operation. Despite this, **existing implementations do not efficiently utilize GPUs**. Even optimized implementations such as Megatron [18] report achieving only 30% of peak GPU flop/s.

We find that **the key bottleneck when training transform-**

.LG] 2 Jul 2020

## High-Performance Communication

- Use larger clusters (10k+ GPUs)
- Model parallelism
  - Complex **pipeline schemes**
- Optimized networks

**Distribution and Parallelism**

| Data | Pipeline | Operator |
|---|---|---|

# High-Performance I/O for Deep Learning


- **Example: ResNet-50 3.8 Gflop inference, ≈3x for training**
  - ImageNet is 150 GiB for ≈1.3M images → average size 115 kiB, range: 508B - 15MiB
  - MLPerf v2.1 on one H100 - 81k samples/s → 9.3 GiB/s random access → ~50 SSDs / GPU

    *Likely more for problems from scientific computing!*
- **Training on thousands of GPUs may need to manage 10,000s of SSDs**

Near-optimal Pre-Fetching System, aka. **NoPFS**

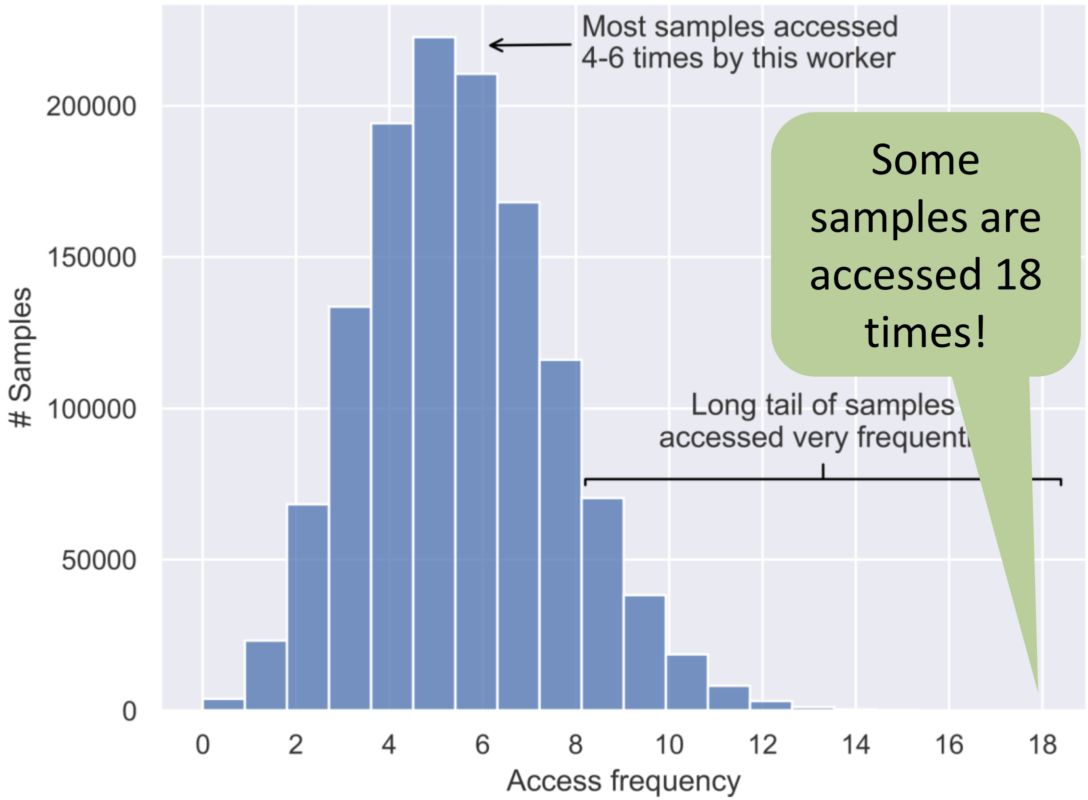
- **But why do we need those even? Deep Learning workloads "randomly sample" input!**
  - By "random", we really mean pseudo-random sequences with fixed seeds ☺

    *This enables clairvoyant prefetching!*

# Clairvoyant Prefetching for Distributed Machine Learning I/O (arXiv 2101.08734)

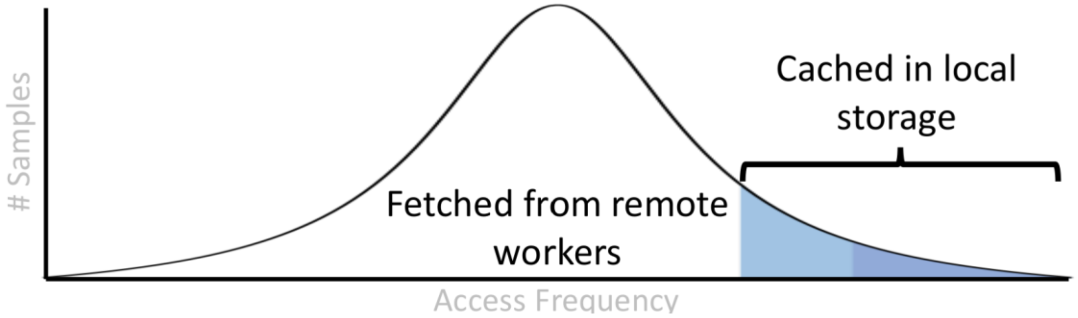
- NoPFS acts as a **distributed cache** – each node keeps cache – fully **knowing about the future**!

single-process access to samples
for ImageNet with 16 processes




PRNG seed → Access stream $R = (\cdots, 7, 4, 5, 8, \cdots)$

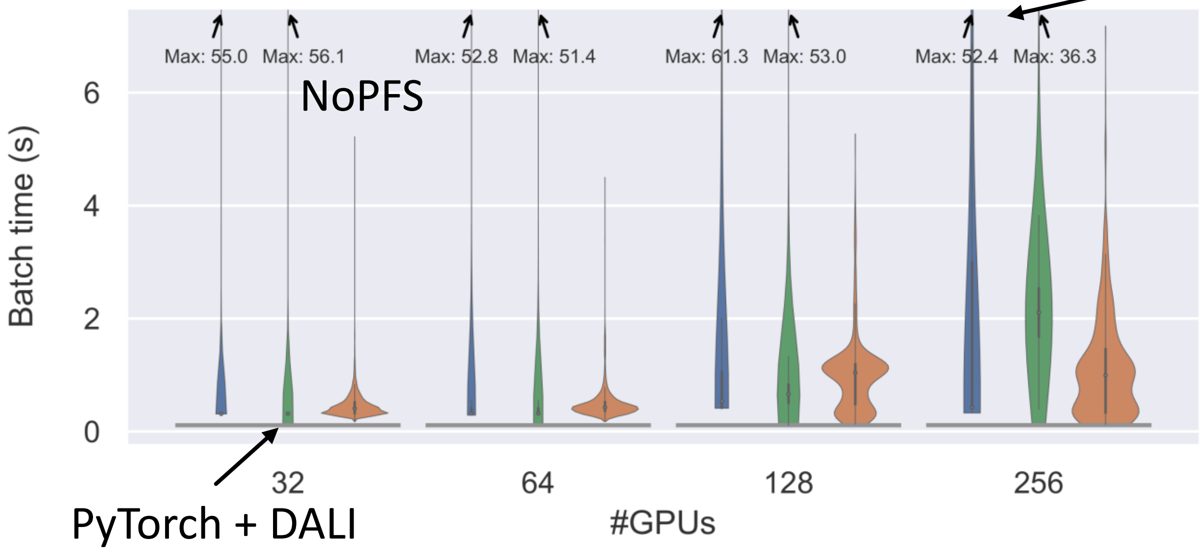
Accesses for worker $i$

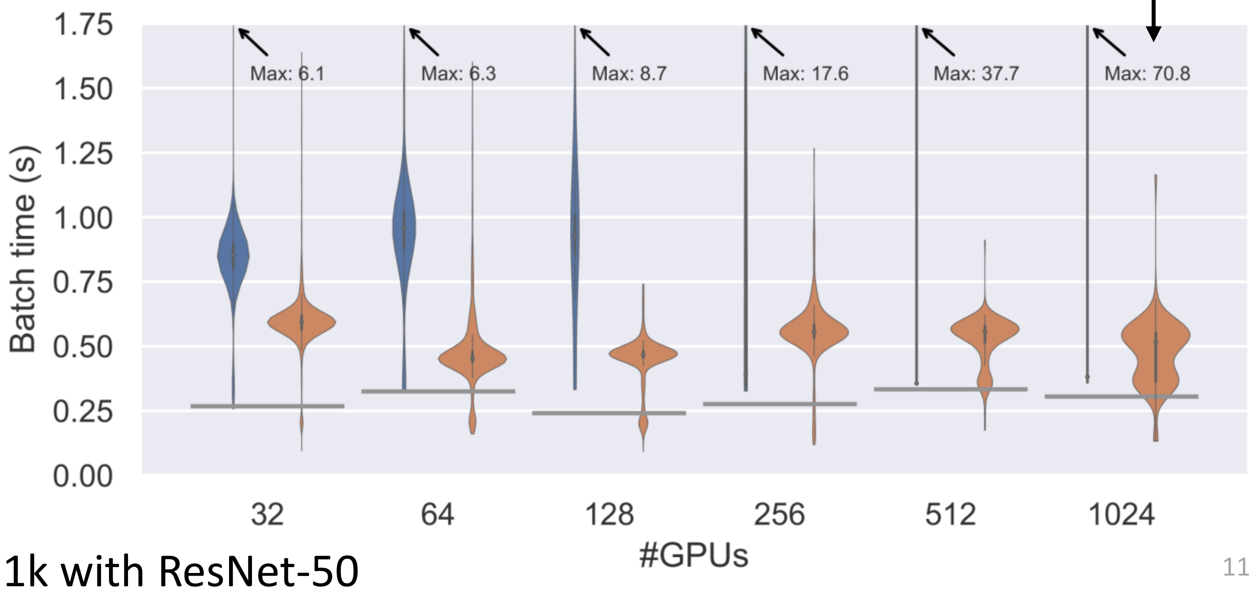


# Clairvoyant Prefetching for Distributed Machine Learning I/O (arXiv 2101.08734)

- NoPFS acts as a **distributed cache** – each node keeps cache – fully **knowing about the future**!





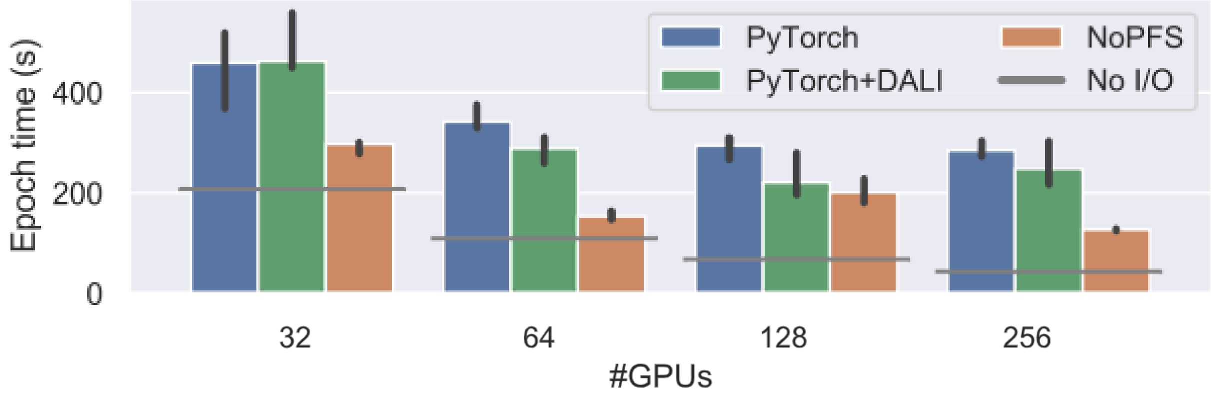

ImageNet 1k with ResNet-50

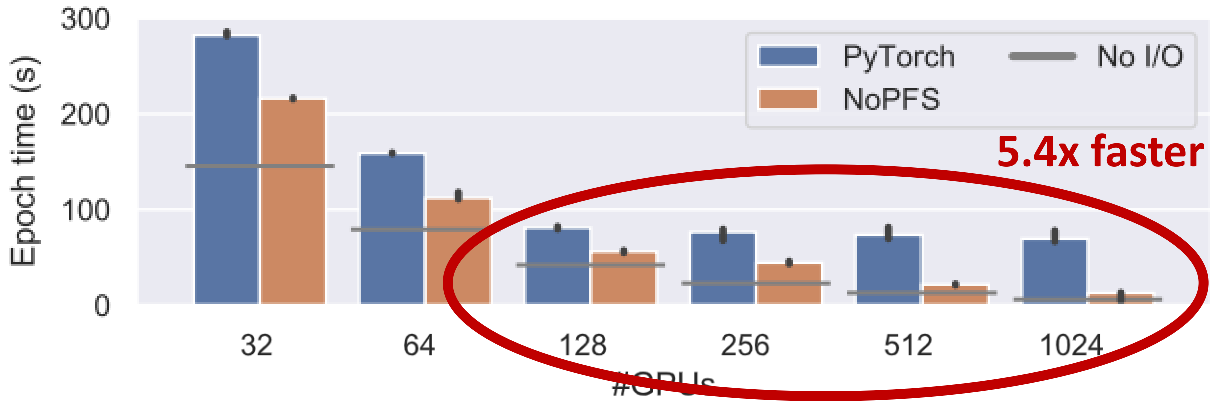
# Clairvoyant Prefetching for Distributed Machine Learning I/O (arXiv 2101.08734)

- NoPFS acts as a distributed cache – each node keeps cache – fully knowing about the future!



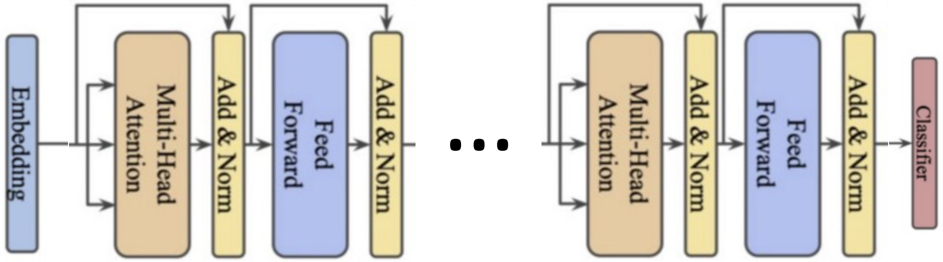
runtime per epoch (full training time)



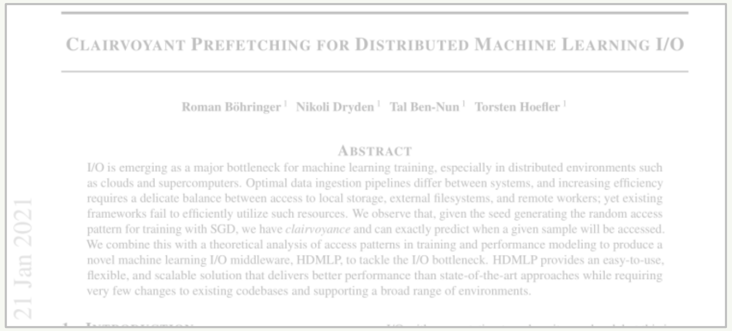

ImageNet 1k with ResNet-50

# Three Systems Dimensions in Large-scale Super-learning …




## High-Performance I/O

- Quickly **growing data volumes**
  - Scientific computing!
- Use the specifics of machine learning workloads
  - E.g., **intelligent prefetching**



CLAIRVOYANT PREFETCHING FOR DISTRIBUTED MACHINE LEARNING I/O

Roman Böhringer[1] Nikoli Dryden[1] Tal Ben-Nun[1] Torsten Hoefler[1]

ABSTRACT

I/O is emerging as a major bottleneck for machine learning training, especially in distributed environments such as clouds and supercomputers. Optimal data ingestion pipelines differ between systems, and increasing efficiency requires a delicate balance between access to local storage, external filesystems, and remote workers; yet existing frameworks fail to efficiently utilize such resources. We observe that, given the seed generating the random access pattern for training with SGD, we have *clairvoyance* and can exactly predict when a given sample will be accessed. We combine this with a theoretical analysis of access patterns in training and performance modeling to produce a novel machine learning I/O middleware, HDMLP, to tackle the I/O bottleneck. HDMLP provides an easy-to-use, flexible, and scalable solution that delivers better performance than state-of-the-art approaches while requiring very few changes to existing codebases and supporting a broad range of environments.

21 Jan 2021

## High-Performance Compute

- Deep learning is HPC
  - **Data movement!**
- **Quantization, Sparsification**
  - Drives modern accelerators!

Data Movement Is All You Need: A Case Study on Optimizing Transformers

Andrei Ivanov*, Nikoli Dryden*, Tal Ben-Nun, Shigang Li, Torsten Hoefler
ETH Zürich
firstname.lastname@inf.ethz.ch
* Equal contribution

*Abstract*—**Transformers have become widely used for language modeling and sequence learning tasks, and are one of the most important machine learning workloads today. Training one is a very compute-intensive task, often taking days or weeks, and significant attention has been given to optimizing transformers. Despite this, existing implementations do not efficiently utilize GPUs. We find that data movement is the key bottleneck when training. Due to Amdahl's Law and massive improvements in compute performance, training has now become memory-bound. Further, existing frameworks use suboptimal data layouts. Using these insights, we present a recipe for globally optimizing data movement in transformers. We reduce data movement by up to 22.91% and overall achieve a 1.30× performance improvement over state-of-the-art frameworks when training BERT. Our approach is applicable more broadly to optimizing deep neural networks, and offers insight into how to tackle emerging performance bottlenecks.**

challenges such as artificial general intelligence [27]. Thus, improving transformer performance has been in the focus of numerous research and industrial groups.

Significant attention has been given to optimizing transformers: local and fixed-window attention [28]–[32], more general structured sparsity [33], learned sparsity [34]–[36], and other algorithmic techniques [19], [37] improve the performance of transformers. Major hardware efforts, such as Tensor Cores and TPUs [38] have accelerated tensor operations like matrix-matrix multiplication (MMM), a core transformer operation. Despite this, **existing implementations do not efficiently utilize GPUs.** Even optimized implementations such as Megatron [18] report achieving only 30% of peak GPU flop/s.

We find that **the key bottleneck when training transform-**

[cs.LG] 2 Jul 2020

## High-Performance Communication

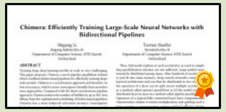
- Use larger clusters (10k+ GPUs)
- Model parallelism
  - Complex **pipeline schemes**
- Optimized networks

**Distribution and Parallelism**

| Data | Pipeline | Operator |
|---|---|---|

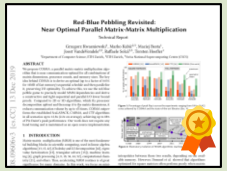
SPARCML: High-Performance Sparse Communication for Machine Learning

Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis



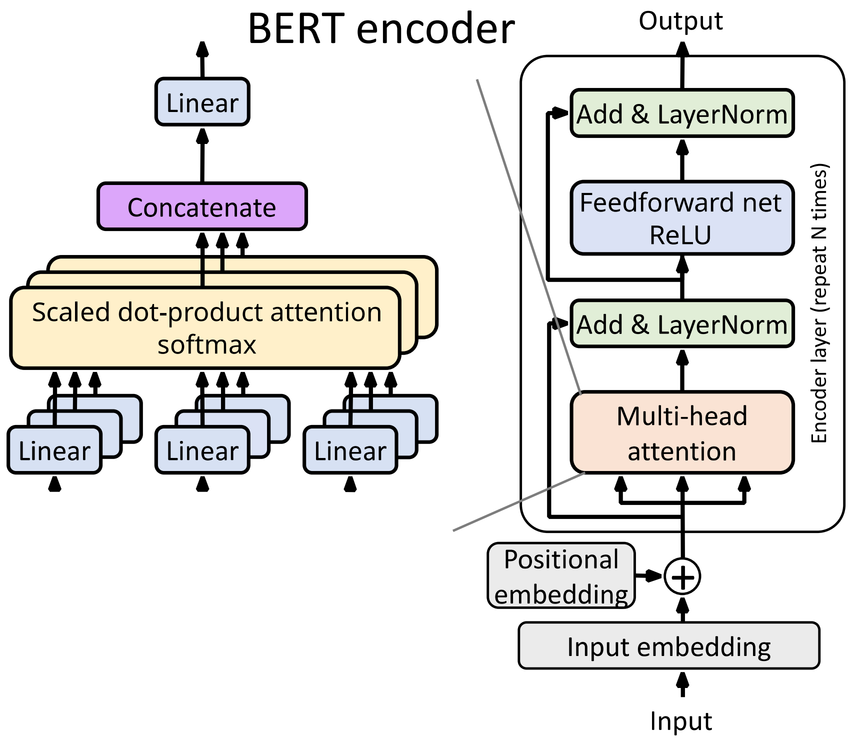
Chimera: Efficiently Training Large-Scale Neural Networks with Bidirectional Pipelines

PipeFisher: Efficient Training of Large Language Models Using Pipelining and Fisher Information Matrices



Red-Blue Pebbling Revisited: Near Optimal Parallel Matrix-Matrix Multiplication

# Data Movement Is All You Need: A Case Study on Optimizing Transformers (arXiv:2007.00072)




highly optimized

| Operator class | % flop | % Runtime |
|---|---|---|
| Tensor contraction | 99.80 | 61.0 |
| Statistical normalization | 0.17 | 25.5 |
| Element-wise | 0.03 | 13.5 |
| | 0.2% | 39% |

OpenAI booth at NeurIPS 2019 in Vancouver, Canada
Image Credit: Khari Johnson / VentureBeat

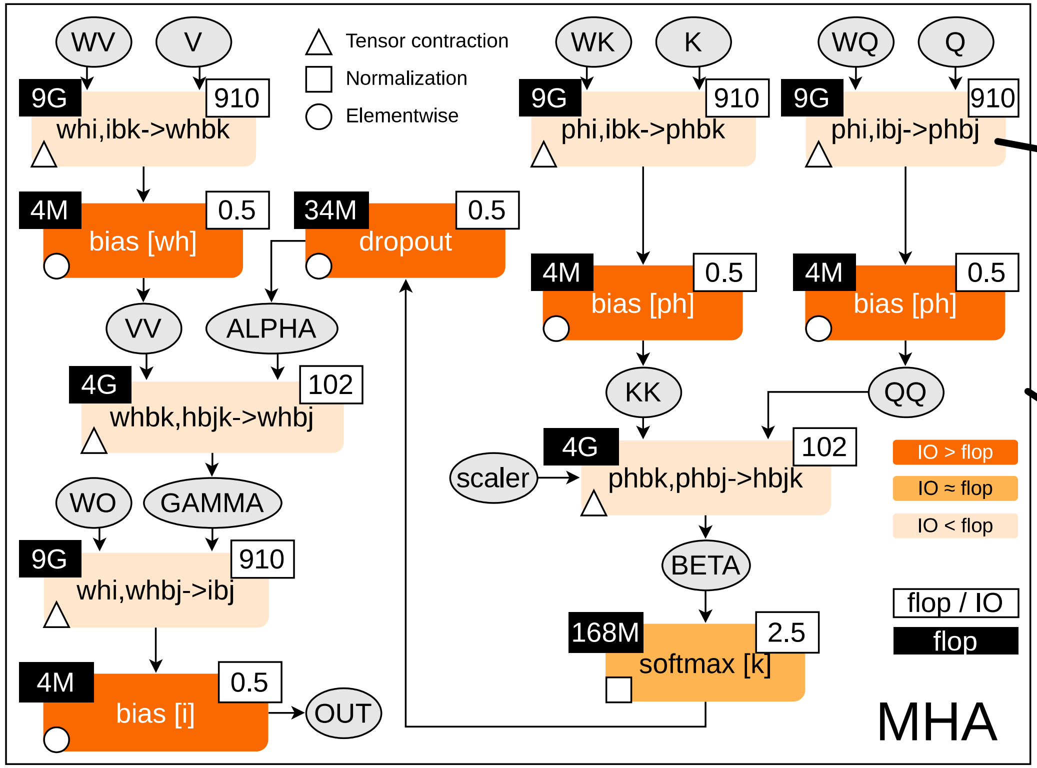
Last week, OpenAI published a paper detailing GPT-3, a machine learning model that achieves strong results on a number of natural language benchmarks. At 175 billion parameters, where a parameter affects data's prominence in an overall prediction, it's the largest of its kind. And with a memory size exceeding 350GB, it's one of the priciest, costing an estimated $12 million to train.

**Our performance improvement for BERT-large**

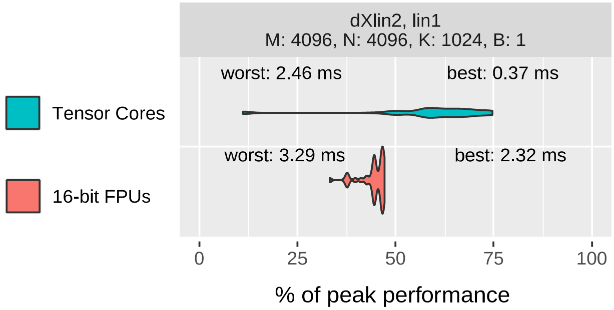
- 30% over PyTorch
- 20% over Tensorflow + XLA
- 8% over DeepSpeed

est. savings on AWS over PyTorch:
$85k for BERT, $3.6M GPT-3

# Data Movement Is All You Need: A Case Study on Optimizing Transformers (arXiv:2007.00072)

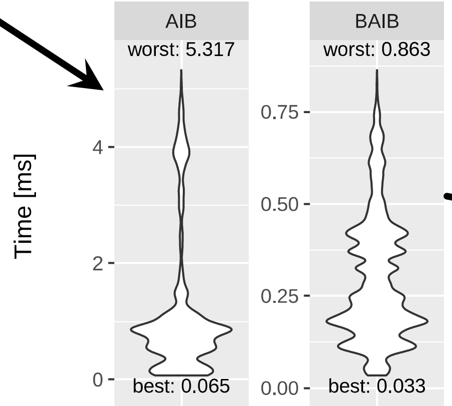



different data layouts

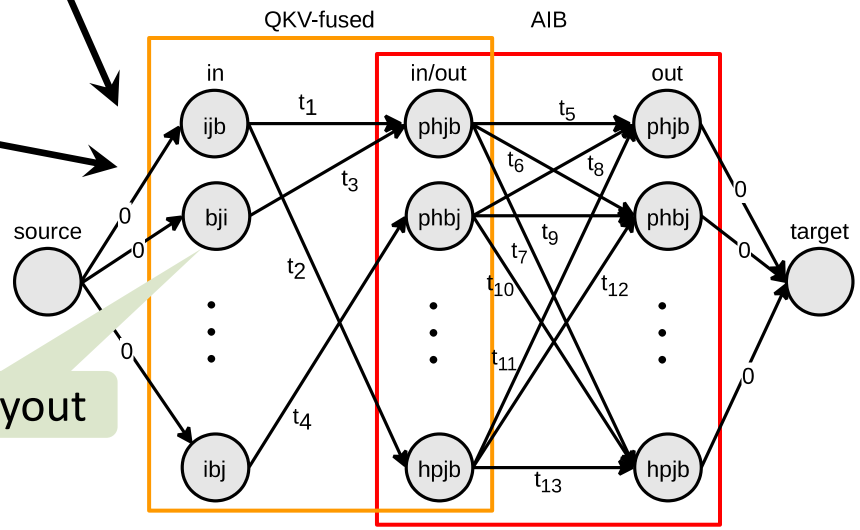



https://github.com/spcl/dace

different fusion strategies




Configuration selection graph




data layout

fusion strategy

Full BERT encoder layer performance (ms)

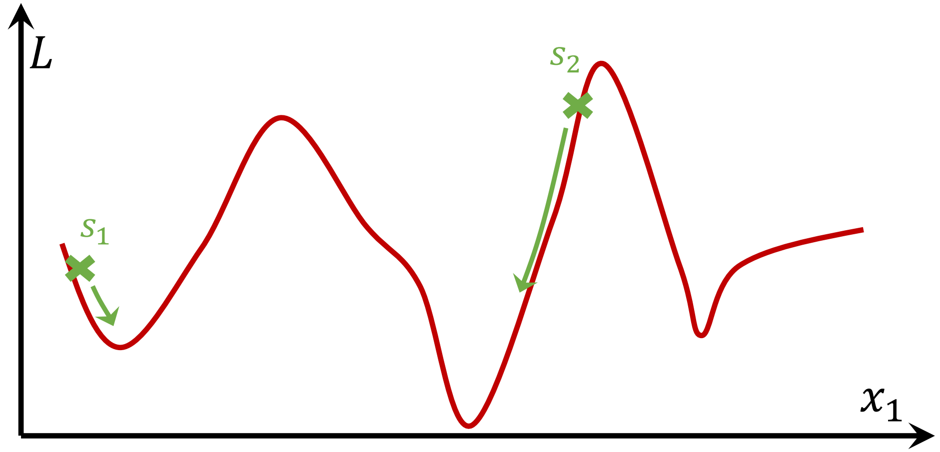
| | TF+XLA | PyTorch | DeepSpeed | Ours |
|---|---|---|---|---|
| Forward | 3.2 | 3.45 | 2.8 | **2.63** |
| Backward | 5.2 | 5.69 | 4.8 | **4.38** |

## Moving Data is Most Expensive!

# Techniques to Shrink ML Data

# Quantization – Running Gigantic LLMs on Reasonable Systems (arXiv:2210.17323)

- **Brains have limited precision! Why are we computing with FP32?**
  - For technical reasons (SGD, optimization, how we quantize)
  - Neurons in Hippocampus can "reliably distinguish 24 strengths" [1]
    *4.6 bits of information!*
- **GPT-3 has up to 175 billion parameters**
  - 700 GiB in FP32, 350 GiB in FP16/BF16 ☹
  - Rounding to <5 bits is not so simple
  - Requires some foundation and many tricks
- **Consider "error landscape" of a trained model with weights $w$ [2]**

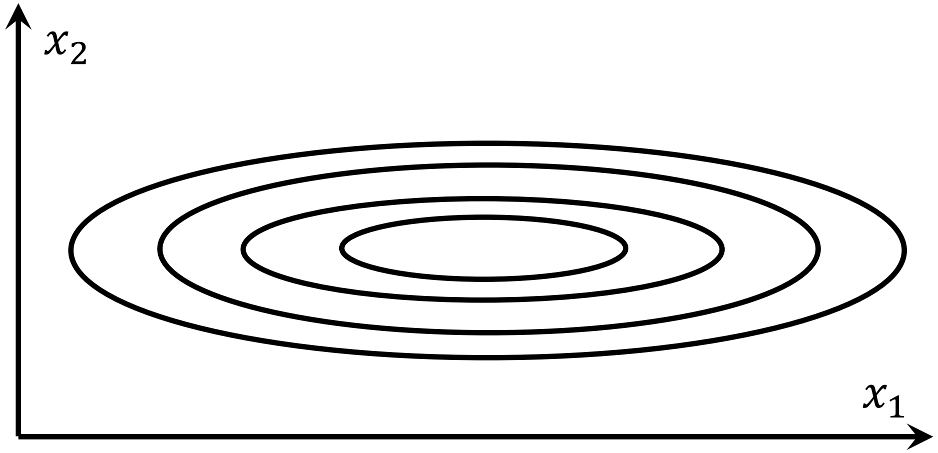



$$\partial E = \left(\frac{\partial E}{\partial w}\right)^T \partial w + \frac{1}{2}\partial w^T \left(\frac{\partial^2 E}{\partial^2 w}\right) \partial w + O(|\partial w|^3)$$

Gradient (≈0)

"Curvature" of error (aka. "sensitivity")

Higher-order terms (=0 for quadratic loss)




[1] Bartol et al., "Hippocampal Spine Head Sizes Are Highly Precise", eLife 2015
[2] LeCun, Denker, Solla: "Optimal Brain Damage", NIPS'90

# Quantization – Running Gigantic LLMs on Reasonable Systems (arXiv:2210.17323)

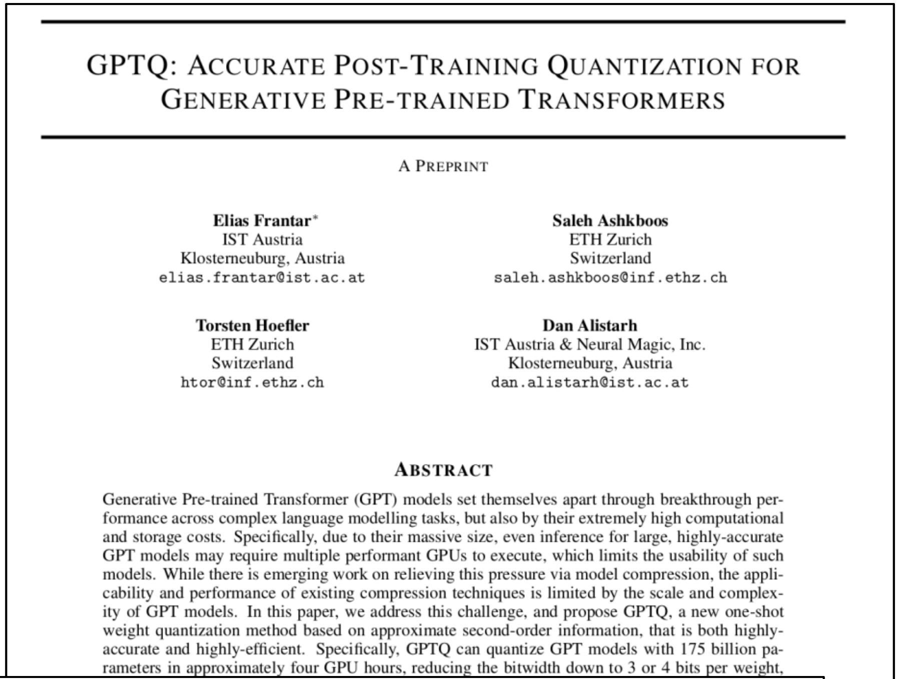
- **Quantization objective for low precision rounded weights $\hat{w}$**

  $$\operatorname{argmin}_{\hat{w}} \|wx - \hat{w}x\|^2$$

- **Solve PTQ optimization problem row by row of $w$**
  - Round row and push the error forward using the inverse Hessian
  - Update Hessian for each column
- **Tricks**
  - Block updates for better locality (10x speedup)
  - Use Cholesky to invert Hessian (higher stability)
  - Work one transformer block at a time (6 operators fit in memory)
  - Use quantized input from previous blocks for block $i$
- **Results**
  - Generative inference 2-4x faster
  - 3 bits → 66 GiB, fits in a single (high-end) A100 GPU!

| Model | FP16 | 1024 | 512 | 256 | 128 | 64 | 32 | 3-bit |
|---|---|---|---|---|---|---|---|---|
| OPT-175B | 8.34 | 11.84 | 10.85 | 10.00 | 9.58 | 9.18 | 8.94 | 8.68 |
| BLOOM | 8.11 | 11.80 | 10.84 | 10.13 | 9.55 | 9.17 | 8.83 | 8.64 |

Table 6: 2-bit GPTQ quantization results with varying group-sizes; perplexity on WikiText2.



GPTQ: ACCURATE POST-TRAINING QUANTIZATION FOR GENERATIVE PRE-TRAINED TRANSFORMERS

A PREPRINT

**Elias Frantar**[*]
IST Austria
Klosterneuburg, Austria
elias.frantar@ist.ac.at

**Saleh Ashkboos**
ETH Zurich
Switzerland
saleh.ashkboos@inf.ethz.ch

**Torsten Hoefler**
ETH Zurich
Switzerland
htor@inf.ethz.ch

**Dan Alistarh**
IST Austria & Neural Magic, Inc.
Klosterneuburg, Austria
dan.alistarh@ist.ac.at

ABSTRACT

Generative Pre-trained Transformer (GPT) models set themselves apart through breakthrough performance across complex language modelling tasks, but also by their extremely high computational and storage costs. Specifically, due to their massive size, even inference for large, highly-accurate GPT models may require multiple performant GPUs to execute, which limits the usability of such models. While there is emerging work on relieving this pressure via model compression, the applicability and performance of existing compression techniques is limited by the scale and complexity of GPT models. In this paper, we address this challenge, and propose GPTQ, a new one-shot weight quantization method based on approximate second-order information, that is both highly-accurate and highly-efficient. Specifically, GPTQ can quantize GPT models with 175 billion parameters in approximately four GPU hours, reducing the bitwidth down to 3 or 4 bits per weight,

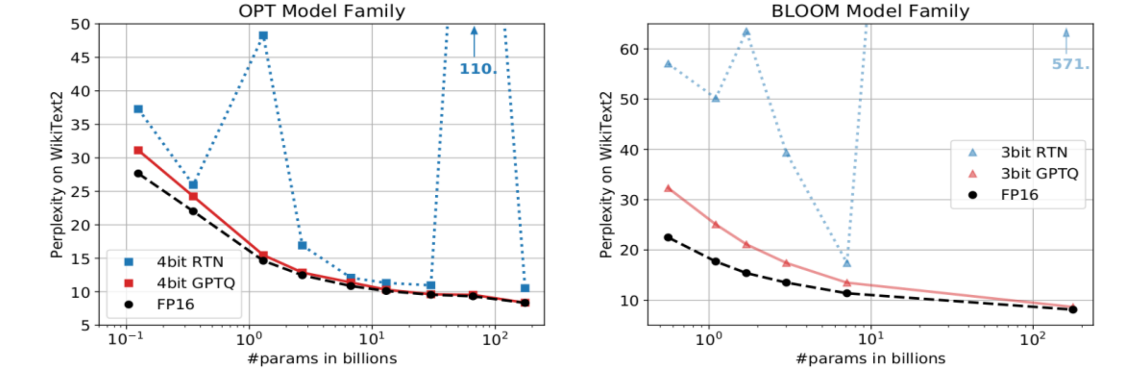
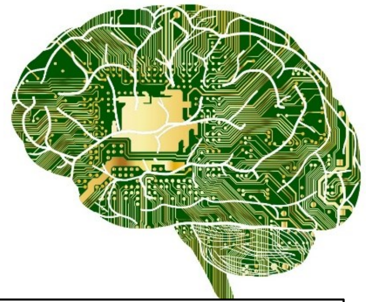



Figure 1: Quantizing OPT models to 4 and BLOOM models to 3 bit precision, comparing GPTQ with the FP16 baseline and round-to-nearest (RTN) [34, 5].

# Quantization Reduces Data by an Order of Magnitude

# How to Go Further?

# Model Sparsification … (arXiv:2102.00554)

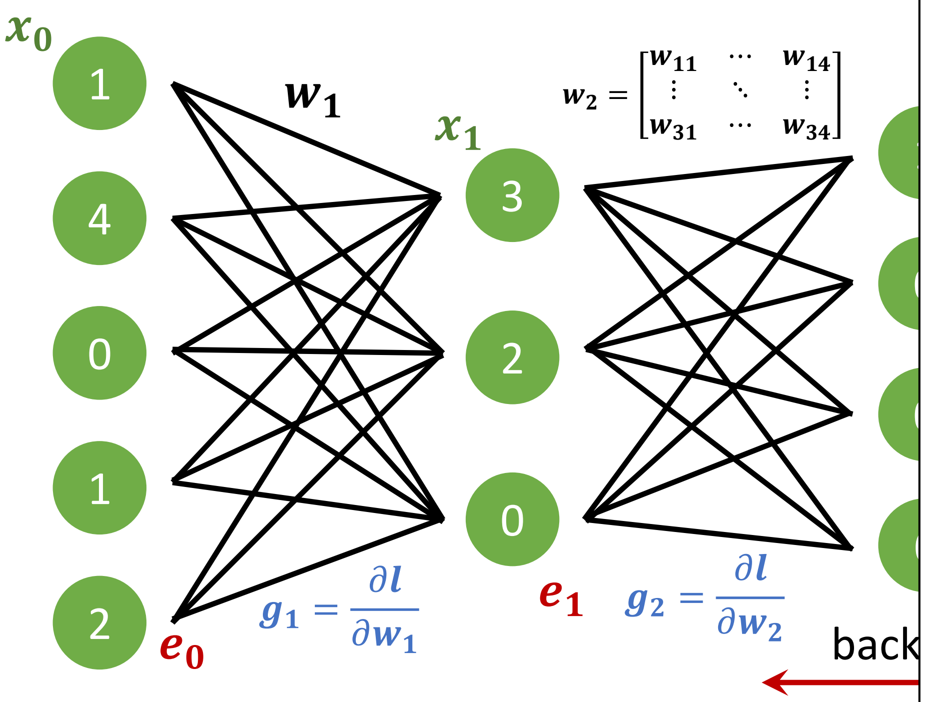


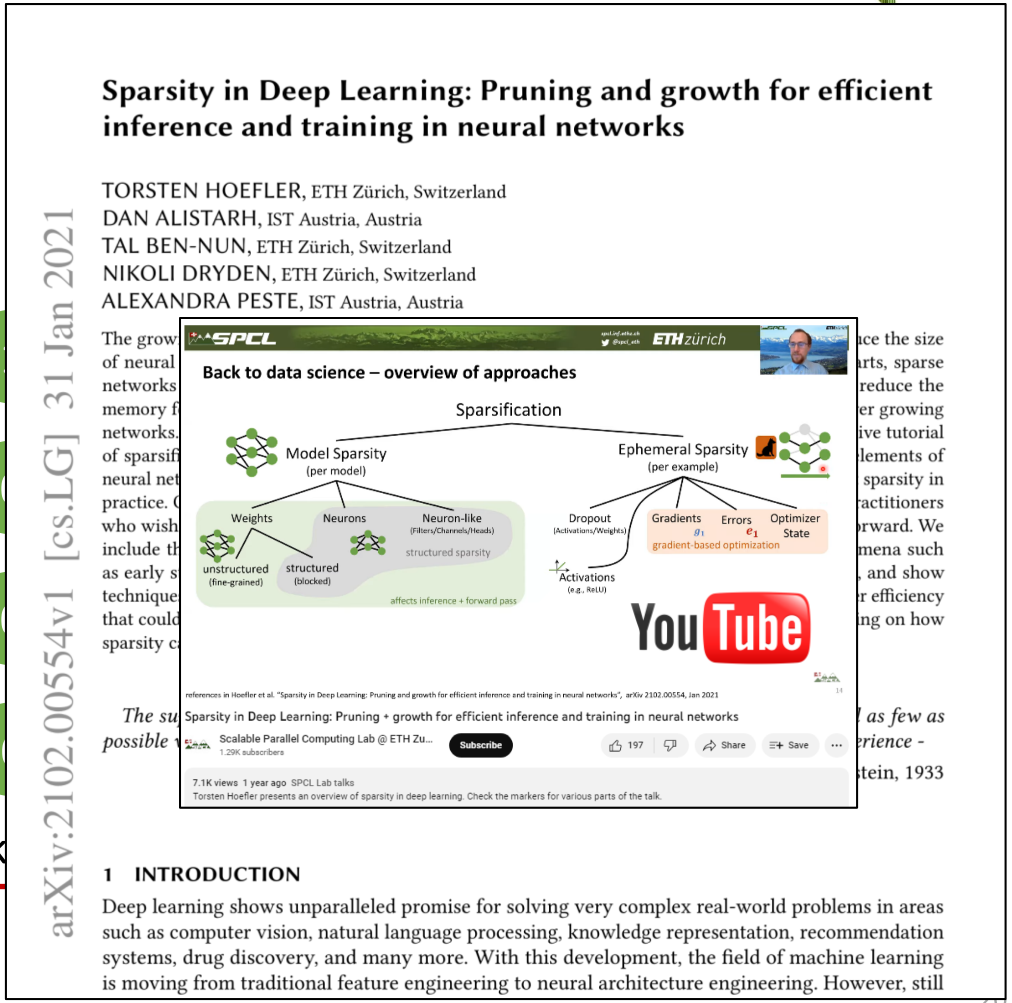
- **Brains are not densely connected! Why are DNN computations dense?**
  - For technical reasons (training, implementation etc.)
  - We may want to shift towards sparse!

Intuition: not all features are always relevant!
- ○ Represent as (sparse) vector space
- ✓ Less overfitting
- ✓ Interpretability
- ✓ Parsimony

**the f_t_re wi_l b_ sp_rs_**

Key results:
- 95% sparse ResNet-52, BERT, or GPT models
- Essentially same quality
- Up to 20x cheaper!





**Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks**

TORSTEN HOEFLER, ETH Zürich, Switzerland
DAN ALISTARH, IST Austria, Austria
TAL BEN-NUN, ETH Zürich, Switzerland
NIKOLI DRYDEN, ETH Zürich, Switzerland
ALEXANDRA PESTE, IST Austria, Austria

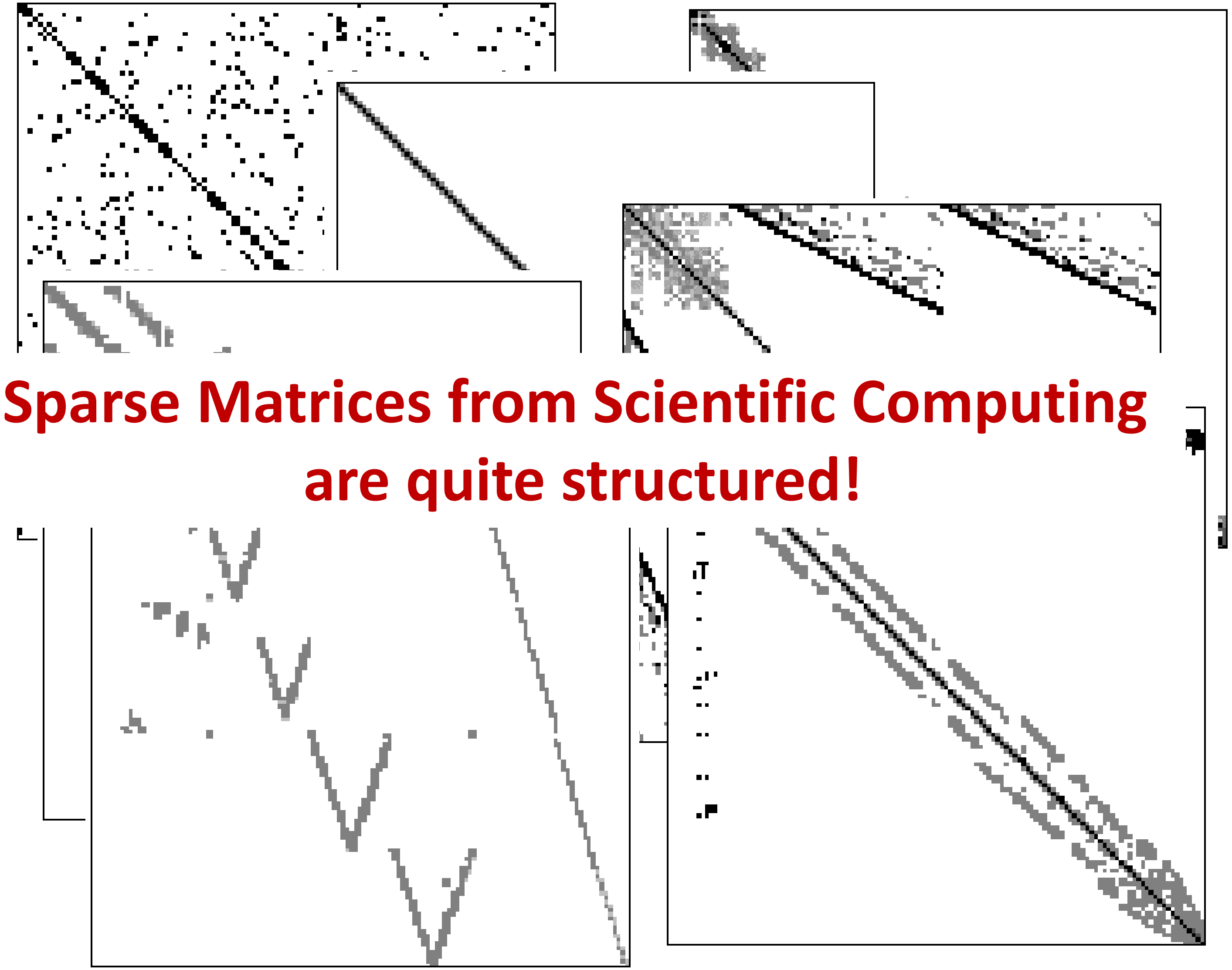
arXiv:2102.00554v1 [cs.LG] 31 Jan 2021



1 INTRODUCTION
Deep learning shows unparalleled promise for solving very complex real-world problems in areas such as computer vision, natural language processing, knowledge representation, recommendation systems, drug discovery, and many more. With this development, the field of machine learning is moving from traditional feature engineering to neural architecture engineering. However, still

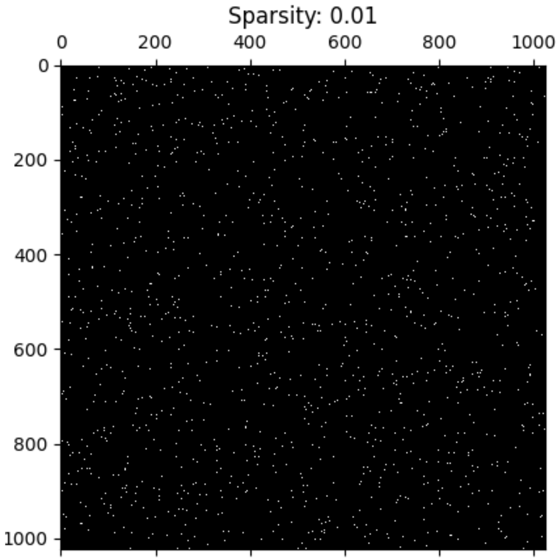
Hoefler et al. "Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks", arXiv 2102.00554, Jan 2021

# Sparse ML Computations – Very Different from Scientific Computing!



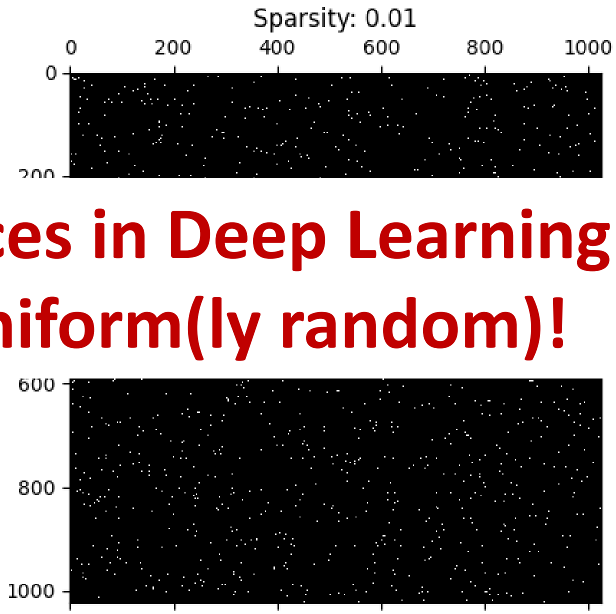
**Sparse Matrices from Scientific Computing are quite structured!**

Source: Suite Sparse (sparse.tamu.edu)




WK

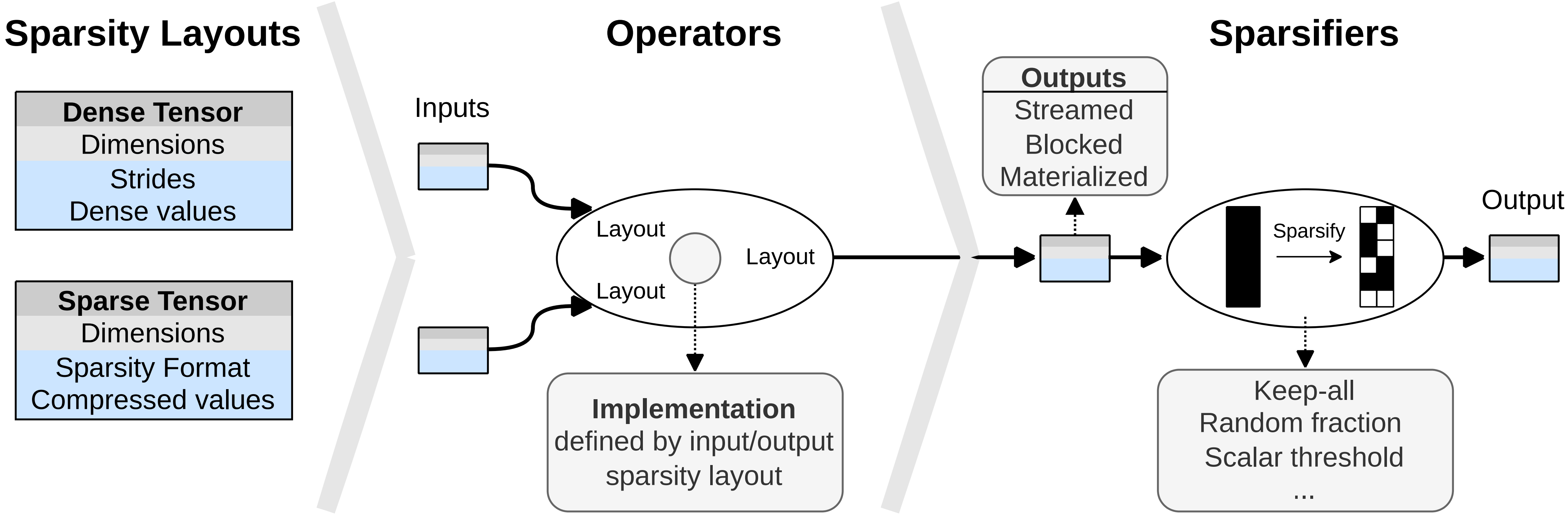
Sparsified BERT
WK and WQ matrices
(3rd encoder)



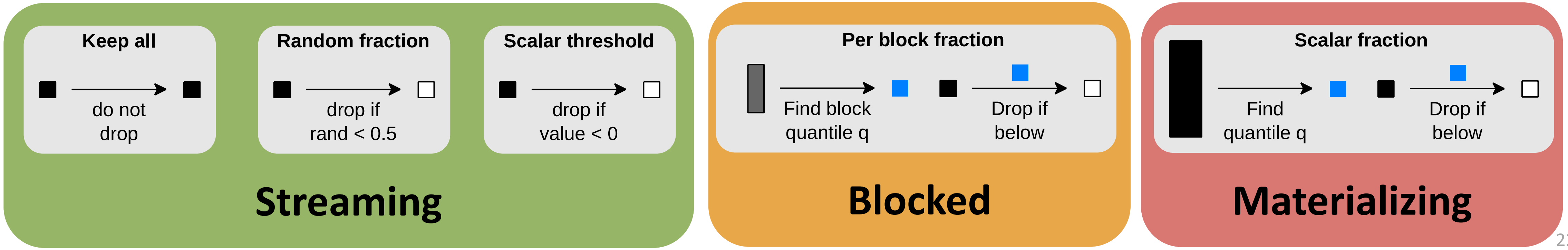

WQ

**Sparse Matrices in Deep Learning are quite uniform(ly random)!**

# Programming Sparse Models – Meet PyTorch Sten (arXiv:2304.07613)



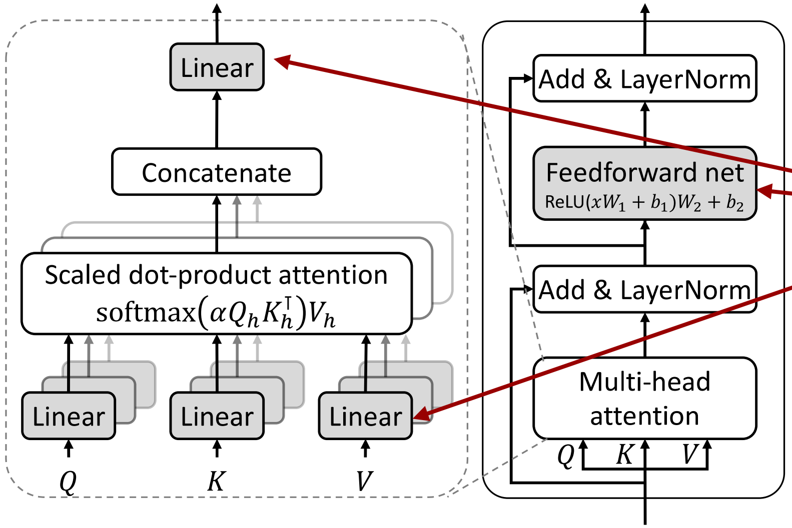

Selected Available Sparsifiers:



# Sten Performance

Custom implementation of matrix multiplication:
**sparse @ dense -> dense**
Linear layer: y = x **W** + b

dense dense **sparse** dense

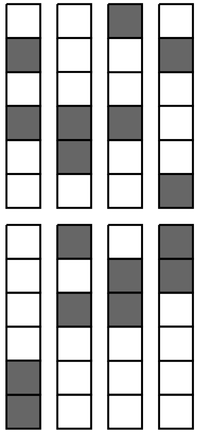


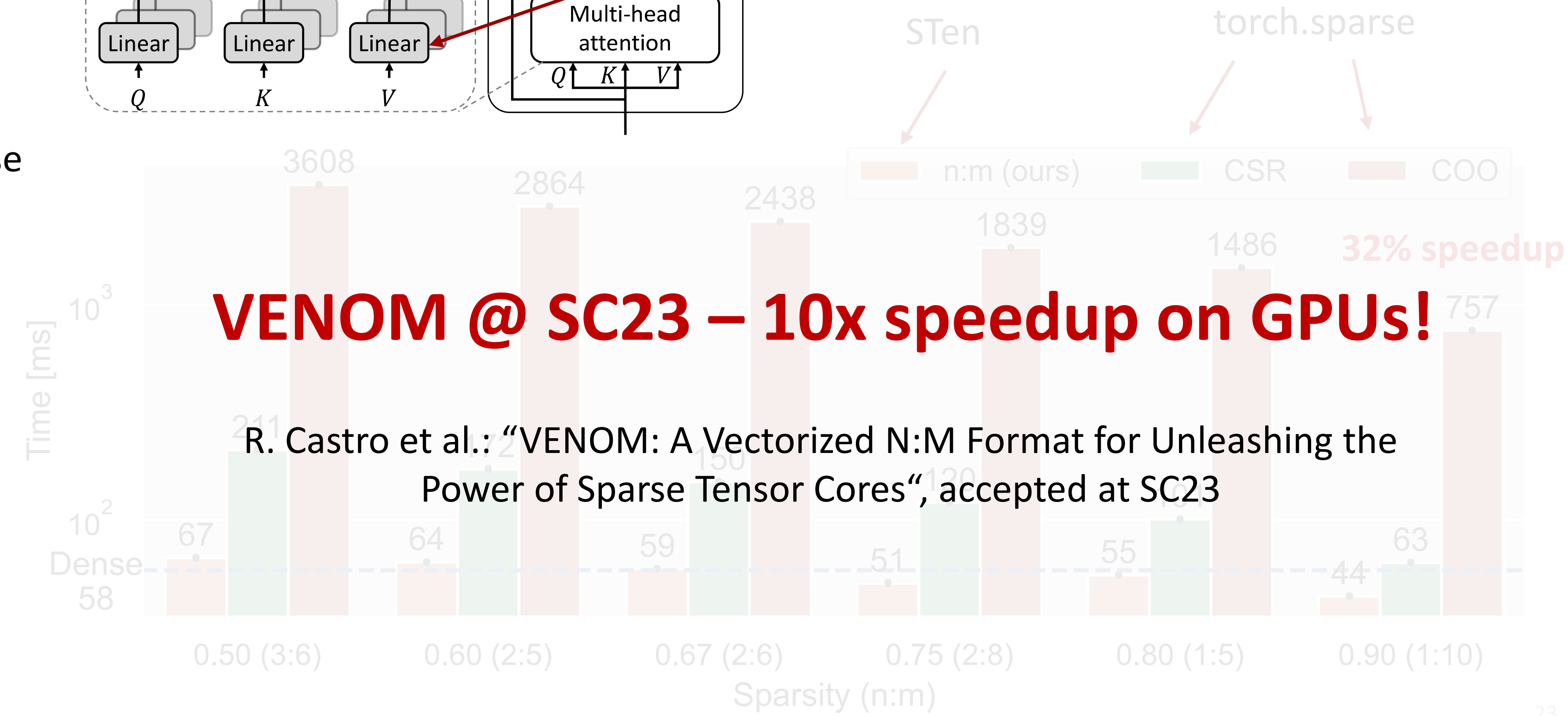

**BERT (base) from HuggingFace**
- batch size 8
- sequence length 128

**Sparsified linear layer weights**

**Intel i7–4770 CPU**

2:6 sparse format






**VENOM @ SC23 – 10x speedup on GPUs!**

R. Castro et al.: “VENOM: A Vectorized N:M Format for Unleashing the Power of Sparse Tensor Cores“, accepted at SC23

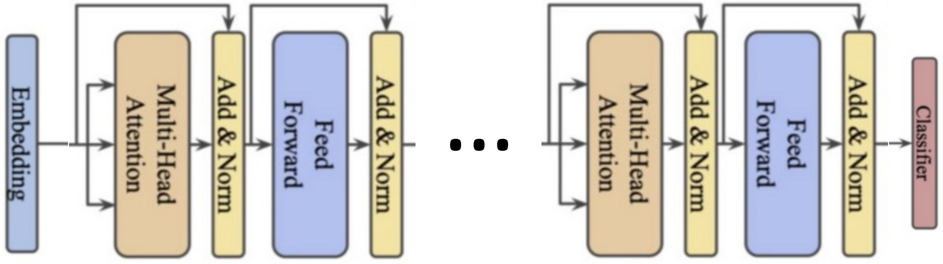
Model Compression Enables

# More Efficient Processing

Which Makes Data Movement Even More Important!

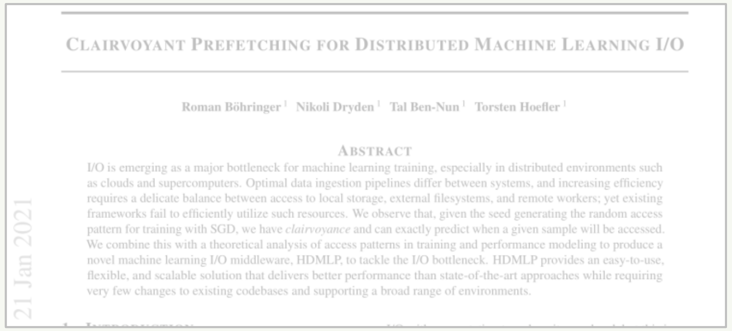
# Especially in the Network!

# Three Systems Dimensions in Large-scale Super-learning …




## High-Performance I/O

- Quickly **growing data volumes**
  - Scientific computing!
- Use the specifics of machine learning workloads
  - E.g., **intelligent prefetching**



CLAIRVOYANT PREFETCHING FOR DISTRIBUTED MACHINE LEARNING I/O

Roman Böhringer [1] Nikoli Dryden [1] Tal Ben-Nun [1] Torsten Hoefler [1]

ABSTRACT

I/O is emerging as a major bottleneck for machine learning training, especially in distributed environments such as clouds and supercomputers. Optimal data ingestion pipelines differ between systems, and increasing efficiency requires a delicate balance between access to local storage, external filesystems, and remote workers; yet existing frameworks fail to efficiently utilize such resources. We observe that, given the seed generating the random access pattern for training with SGD, we have *clairvoyance* and can exactly predict when a given sample will be accessed. We combine this with a theoretical analysis of access patterns in training and performance modeling to produce a novel machine learning I/O middleware, HDMLP, to tackle the I/O bottleneck. HDMLP provides an easy-to-use, flexible, and scalable solution that delivers better performance than state-of-the-art approaches while requiring very few changes to existing codebases and supporting a broad range of environments.

21 Jan 2021

## High-Performance Compute

- Deep learning is HPC
  - **Data movement!**
- **Quantization, Sparsification**
  - Drives modern accelerators!

Data Movement Is All You Need: A Case Study on Optimizing Transformers

Andrei Ivanov*, Nikoli Dryden*, Tal Ben-Nun, Shigang Li, Torsten Hoefler
ETH Zürich
firstname.lastname@inf.ethz.ch
* Equal contribution

*Abstract*—**Transformers have become widely used for language modeling and sequence learning tasks, and are one of the most important machine learning workloads today. Training one is a very compute-intensive task, often taking days or weeks, and significant attention has been given to optimizing transformers. Despite this, existing implementations do not efficiently utilize GPUs. We find that data movement is the key bottleneck when training. Due to Amdahl's Law and massive improvements in compute performance, training has now become memory-bound. Further, existing frameworks use suboptimal data layouts. Using these insights, we present a recipe for globally optimizing data movement in transformers. We reduce data movement by up to 22.91% and overall achieve a 1.30× performance improvement over state-of-the-art frameworks when training BERT. Our approach is applicable more broadly to optimizing deep neural networks, and offers insight into how to tackle emerging performance bottlenecks.**

challenges such as artificial general intelligence [27]. Thus, improving transformer performance has been in the focus of numerous research and industrial groups.

Significant attention has been given to optimizing transformers: local and fixed-window attention [28]–[32], more general structured sparsity [33], learned sparsity [34]–[36], and other algorithmic techniques [19], [37] improve the performance of transformers. Major hardware efforts, such as Tensor Cores and TPUs [38] have accelerated tensor operations like matrix-matrix multiplication (MMM), a core transformer operation. Despite this, **existing implementations do not efficiently utilize GPUs**. Even optimized implementations such as Megatron [18] report achieving only 30% of peak GPU flop/s.

We find that **the key bottleneck when training transform-**

2 Jul 2020

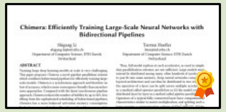
## High-Performance Communication

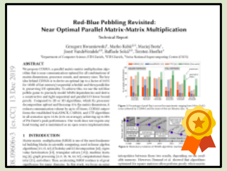
- Use larger clusters (10k+ GPUs)
- Model parallelism
  - Complex **pipeline schemes**
- Optimized networks

**Distribution and Parallelism**

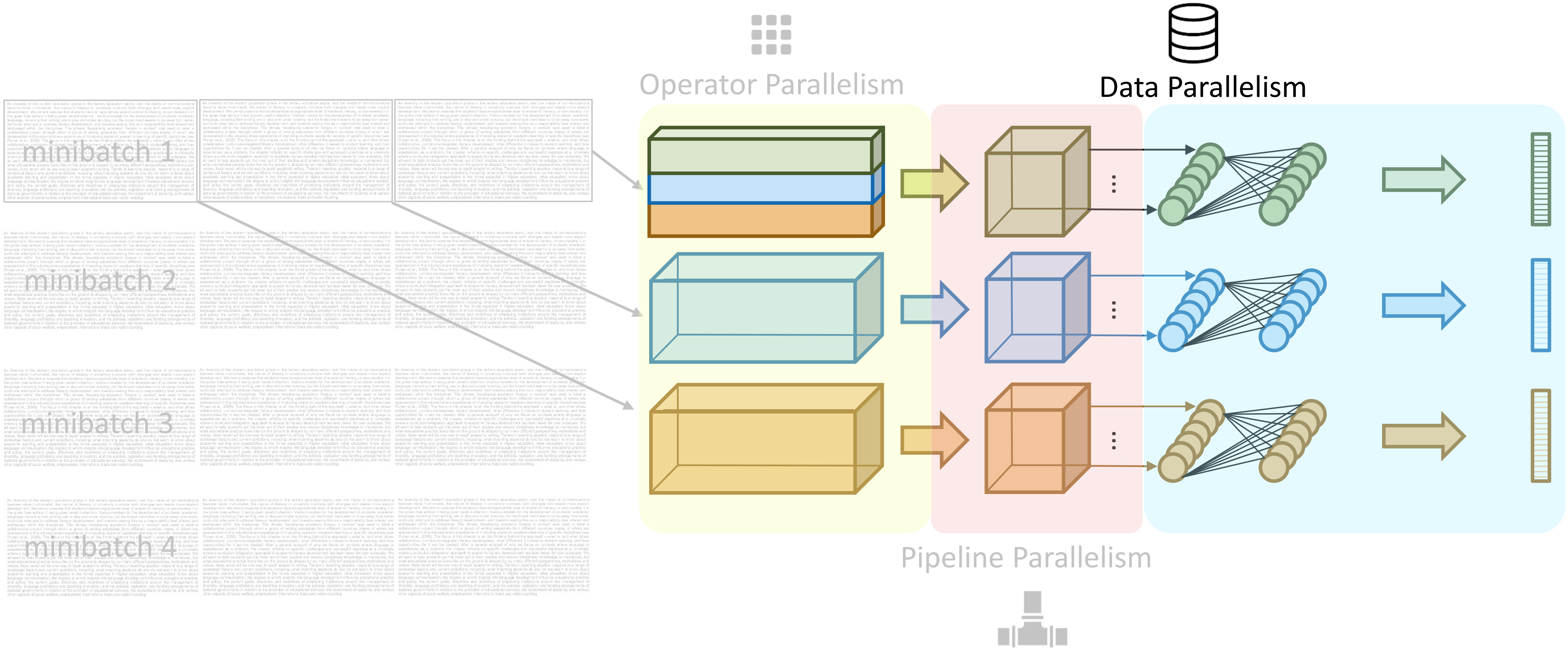
| Data | Pipeline | Operator |
| --- | --- | --- |





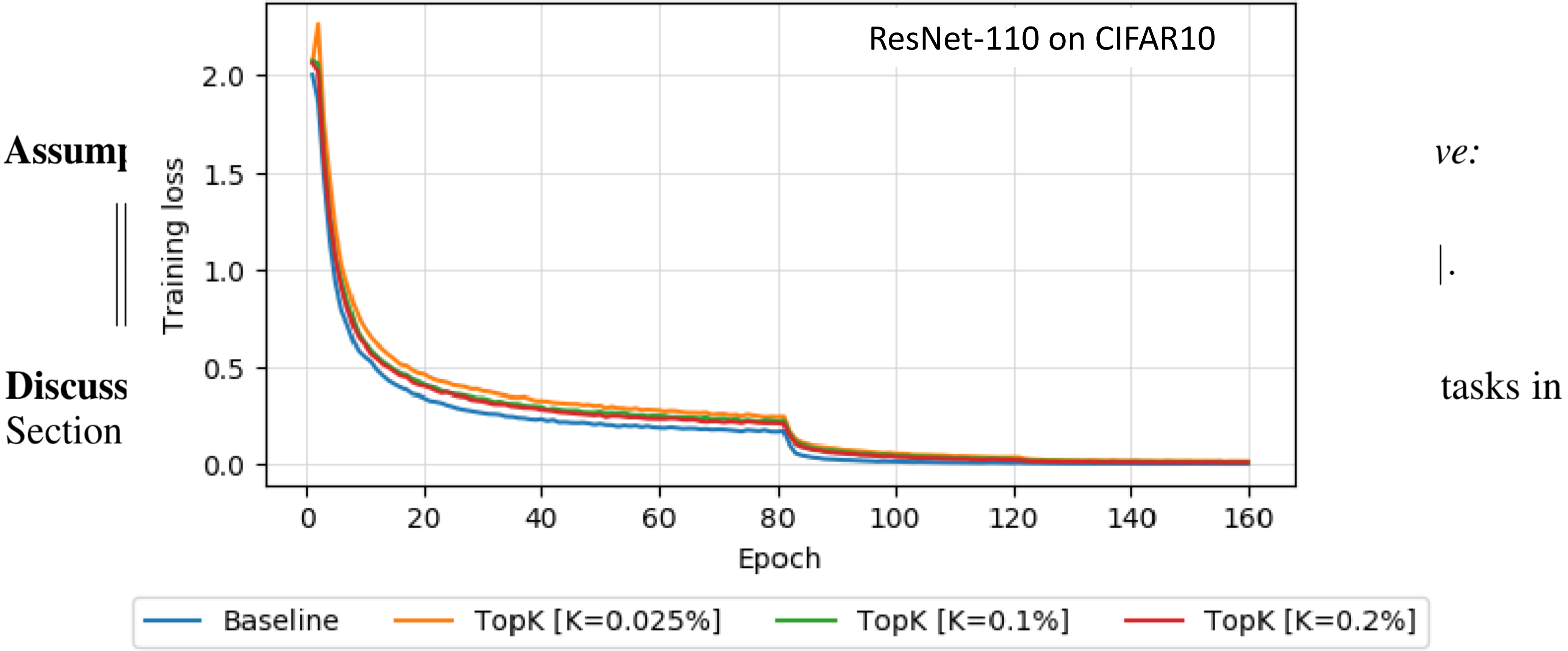
# The Three Dimensions of Parallelism in Deep Learning (arXiv:1802.09941)




T. Ben-Nun, T. Hoefler: Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis, arXiv Feb 2018

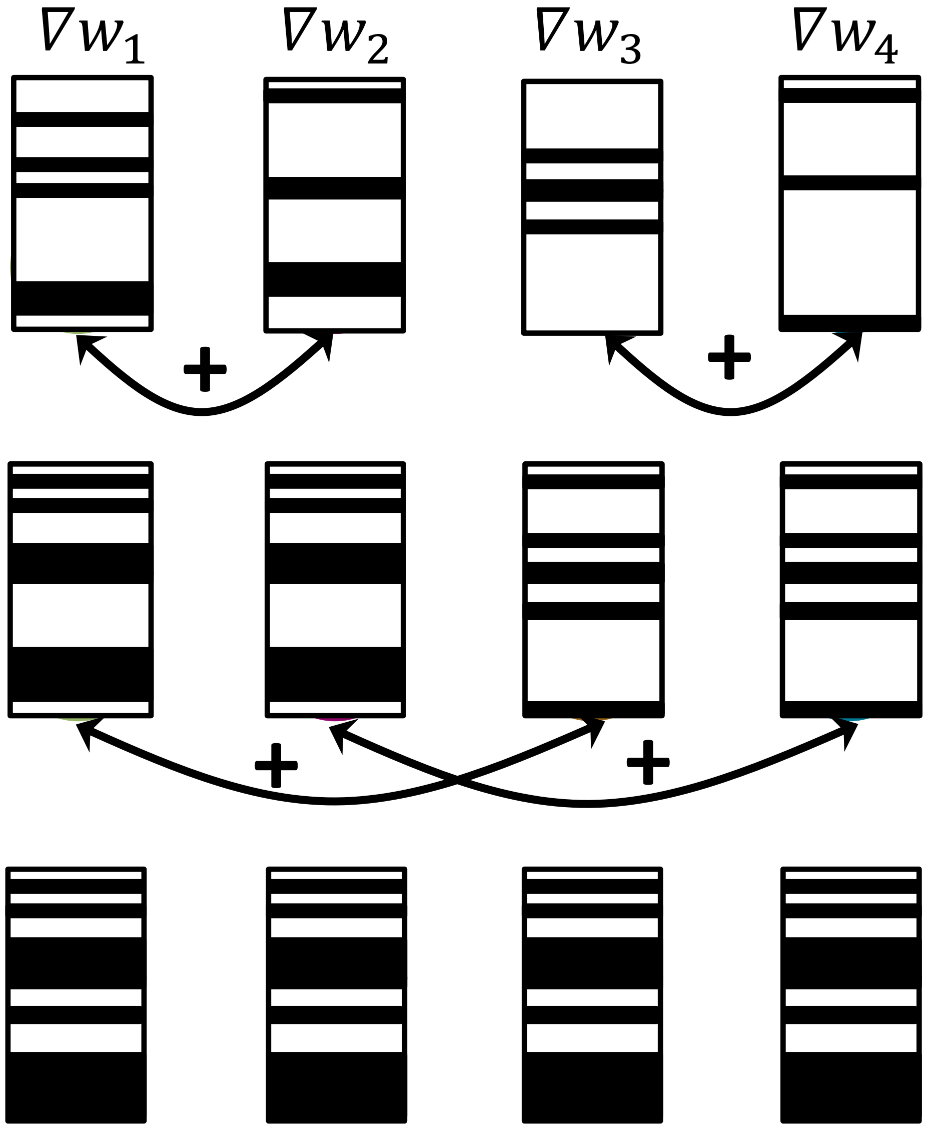
# Data-parallel Gradient Sparsification – Top-k SGD (arXiv:1809.10505)

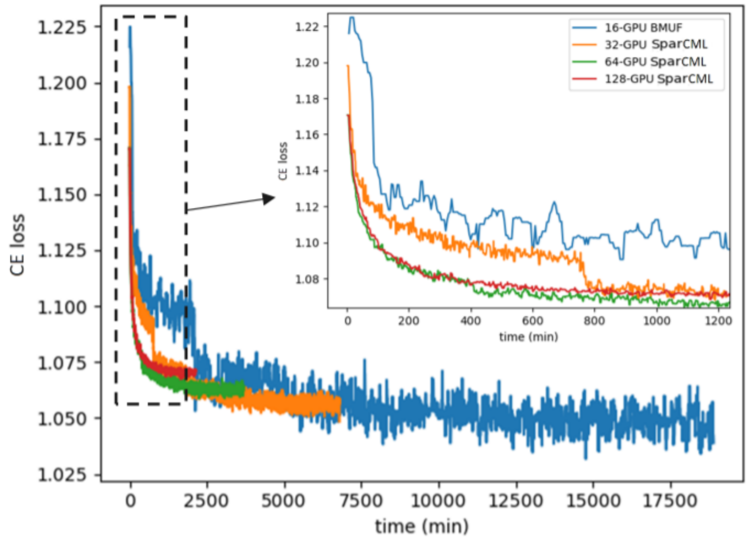
- **Turns out 90-99.9% of the smallest gradient values can be skipped in the summation – at similar accuracy**
  - **Accumulate** the **skipped values locally** (convergence proof, similar to async. SGD with implicit staleness bounds [1])

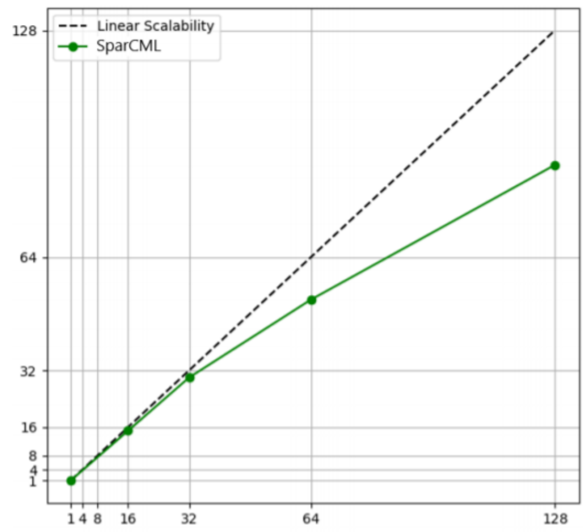



[1] Dan Alistarh, TH, et al.: “The Convergence of Sparsified Gradient Methods”, NIPS’18

# SparCML – Sparse Allreduce for Decentral Updates (arXiv:1802.08021)




Microsoft Speech Production Workload Results – **2 weeks → 2 days**!

| System | Dataset | Model | # of nodes | Algorithm | Speedup |
|---|---|---|---|---|---|
| Piz Daint | ImageNet | VGG19 | 8 | Q4 | 1.55 (3.31) |
| Piz Daint | ImageNet | AlexNet | 16 | Q4 | 1.30 (1.36) |
| Piz Daint<br>EC2 | MNIST | MLP | 8 | Top16_Q4<br>Top16_Q4 | 3.65 (4.53)<br>19.12 (22.97) |

C. Renggli, TH et al. SparCML: High-Performance Sparse Communication for Machine Learning, SC19

# Sparse Allreduce – A Headache for Systems Work

## Near-Optimal Sparse Allreduce for Distributed Deep Learning

Shigang Li
shigang.li@inf.ethz.ch
Department of Computer Science, ETH Zurich
Switzerland

Torsten Hoefler
htor@inf.ethz.ch
Department of Computer Science, ETH Zurich
Switzerland

### Abstract

Communication overhead is one of the major obstacles to train large deep learning models at scale. Gradient sparsification is a promising technique to reduce the communication volume. However, it is very challenging to obtain real performance improvement because of (1) the difficulty of achieving an scalable and efficient sparse *allreduce* algorithm and (2) the sparsification overhead. This paper proposes O*k*-Top*k*, a scheme for distributed training with sparse gradients. O*k*-Top*k* integrates a novel sparse allreduce algorithm (less than 6*k* communication volume which is asymptotically optimal) with the decentralized parallel Stochastic Gradient Descent (SGD) optimizer, and its convergence is proved. To reduce the sparsification overhead, O*k*-Top*k* efficiently selects the top-*k* gradient values according to an estimated threshold. Evaluations are conducted on the Piz Daint supercomputer with neural network models from different deep learning domains. Empirical results show that O*k*-Top*k* achieves similar model accuracy to dense allreduce. Compared with the optimized dense and the state-of-the-art sparse allreduces, O*k*-Top*k* is more scalable and significantly improves training throughput (e.g., 3.29x-12.95x improvement for BERT on 256 GPUs).

*CCS Concepts:* • **Theory of computation → Parallel algorithms**; • **Computing methodologies** → *Neural networks*.

***Keywords:*** distributed deep learning, allreduce, gradient sparsification, data parallelism

introducing up to 99.9% zero values without significant loss of accuracy. Only the nonzero values of the distributed gradients are accumulated across all processes. See [22] for an overview of gradient and other sparsification approaches in deep learning.

However, sparse reductions suffer from scalability issues. Specifically, the communication volume of the existing sparse reduction algorithms grows with the number of processes $P$. Taking the *allgather*-based sparse reduction [36, 41, 47] as an example, its communication volume is proportional to $P$, which eventually surpasses the dense *allreduce* as $P$ increases. Other more complex algorithms [36] suffer from significant fill-in during the reduction, which also leads to a quick increase of the data volume as $P$ grows, and may degrade to dense representations on the fly. For example, let us assume the model has 1 million weights and it is 99% sparse at each node—thus, each node contributes its 10,000 largest gradient values and their indexes to the calculation. Let us now assume that the computation is distributed across 128 data-parallel nodes and the reduction uses a dissemination algorithm [20, 28] with 7 stages. In stage one, each process communicates its 10,000 values to be summed up. Each process now enters the next stage with up to 20,000 values. Those again are summed up leading to up to 40,000 values in stage 3 (if the value indexes do not overlap). The number of values grows exponentially until the algorithm converges after 7 stages with 640,000 values (nearly dense!). Even with overlapping indexes, the fill-in will quickly diminish the benefits of gradient sparsity in practice and lead to large and suboptimal communication volumes [36].

**Minimize fill-in and communication volume!**

## Flare: Flexible In-Network Allreduce

Daniele De Sensi
daniele.desensi@inf.ethz.ch
ETH Zurich
Zurich, Switzerland

Salvatore Di Girolamo
salvatore.digirolamo@inf.ethz.ch
ETH Zurich
Zurich, Switzerland

Saleh Ashkboos
saleh.ashkboos@inf.ethz.ch
ETH Zurich
Zurich, Switzerland

Shigang Li
shigang.li@inf.ethz.ch
ETH Zurich
Zurich, Switzerland

Torsten Hoefler
torsten.hoefler@inf.ethz.ch
ETH Zurich
Zurich, Switzerland

### ABSTRACT

The *allreduce* operation is one of the most commonly used communication routines in distributed applications. To improve its bandwidth and to reduce network traffic, this operation can be accelerated by offloading it to network switches, that aggregate the data received from the hosts, and send them back the aggregated result. However, existing solutions provide limited customization opportunities and might provide suboptimal performance when dealing with custom operators and data types, with sparse data, or when reproducibility of the aggregation is a concern. To deal with these problems, in this work we design a flexible programmable switch by using as a building block PsPIN, a RISC-V architecture implementing the sPIN programming model. We then design, model, and analyze different algorithms for executing the aggregation on this architecture, showing performance improvements compared to state-of-the-art approaches.

### CCS CONCEPTS

• **Networks** → **In-network processing**; • **Hardware** → **Networking hardware**; • **Computer systems organization** → **Distributed architectures**.

### KEYWORDS

In-Network Computing; Programmable Switch; Allreduce

**ACM Reference Format:**
Daniele De Sensi, Salvatore Di Girolamo, Saleh Ashkboos, Shigang Li, and Torsten Hoefler. 2018. Flare: Flexible In-Network Allreduce. In *Supercomputing '21: The International Conference for High Performance Computing, Networking, Storage, and Analysis, Nov 14–19, 2021, St. Louis, MO.* ACM, New

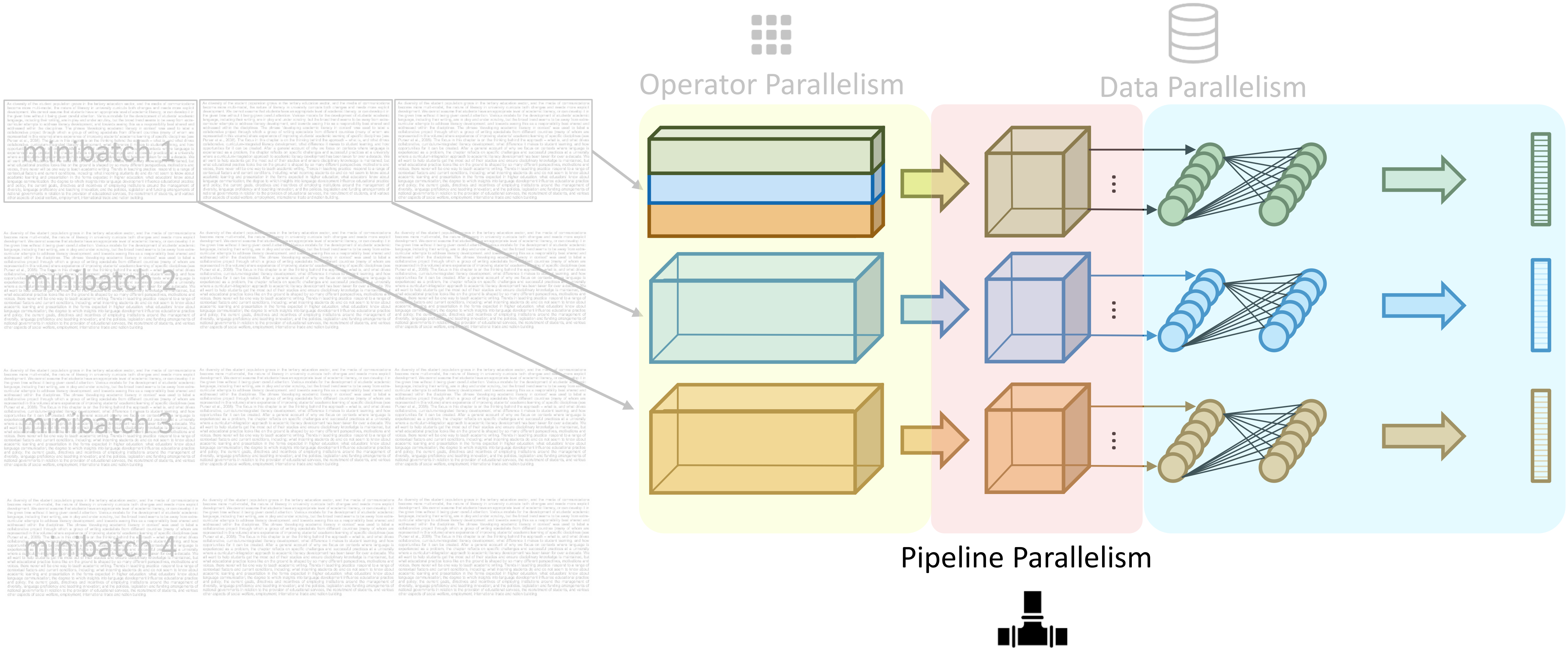
others, and recent studies [2] show that "`MPI_Allreduce` *is the most significant collective in terms of usage and time*".

The simplest bandwidth-optimal allreduce algorithm is the Rabenseifner algorithm (also known as *ring allreduce*) [3]. This algorithm is composed of two phases: a *scatter-reduce* and an *allgather* phase. All hosts are arranged into a logical ring and in each of these two phases, each host sends to its neighbor $(P-1)$ messages, each of size $\frac{Z}{P}$ (where $Z$ is the number of elements to be reduced). The total amount of data sent by each host is then $2(P-1)\frac{Z}{P} \approx 2Z$. To reduce the amount of transmitted data, and thus increase the performance, hosts can exploit *in-network* compute, i.e., they can offload the allreduce operation to the switches in the network.

To outline the advantages of performing an in-network allreduce, we describe the general idea underlying most existing in-network reduction approaches [9–11]. We first suppose to have the $P$ hosts connected through a single switch. Each of the hosts sends its data to the switch, that aggregates together the vectors coming from all the hosts, and then sends them back the aggregated vector. Differently from the host-based optimal allreduce, in the in-network allreduce each host only sends $Z$ elements, thus leading to a 2x reduction in the amount of transmitted data. If the switches can aggregate the received data at line rate, this leads to a **2x bandwidth improvement** compared to a host-based allreduce. Besides improvements in the bandwidth, in-network allreduce also reduces the network traffic. Because the interconnection network consumes a large fraction of the overall system power (from 15% to 50% depending on the system load [12]), any reduction in the network traffic would also help in reducing the power consumption and thus the running cost of the system.

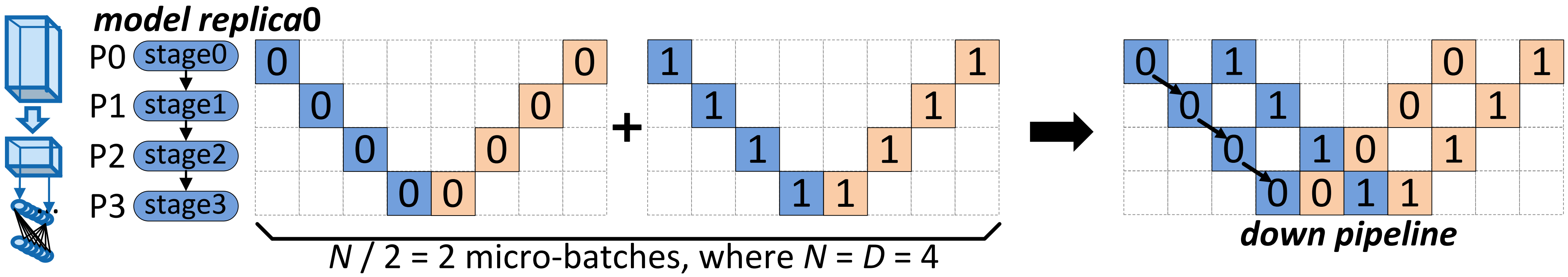
**Making in-network computation work!**

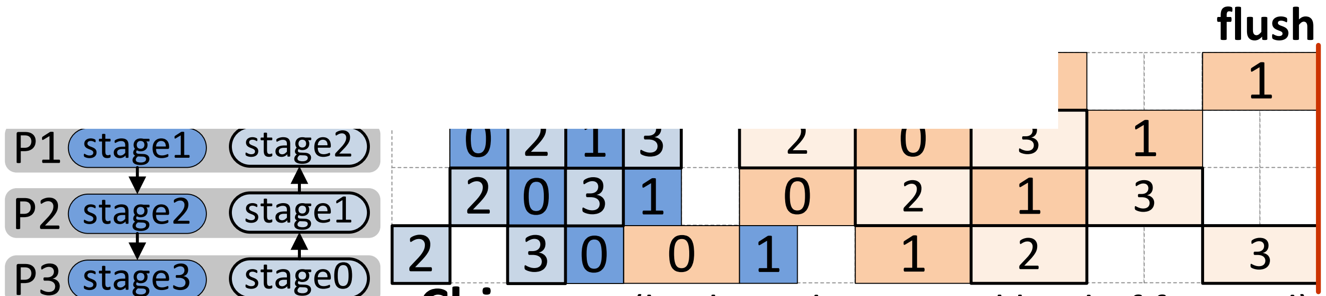
# The Three Dimensions of Parallelism in Deep Learning (arXiv:1802.09941)




T. Ben-Nun, T. Hoefler: Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis, arXiv Feb 2018

# Bidirectional Pipelines – Meet Chimera (arXiv: 2107.06925v3)

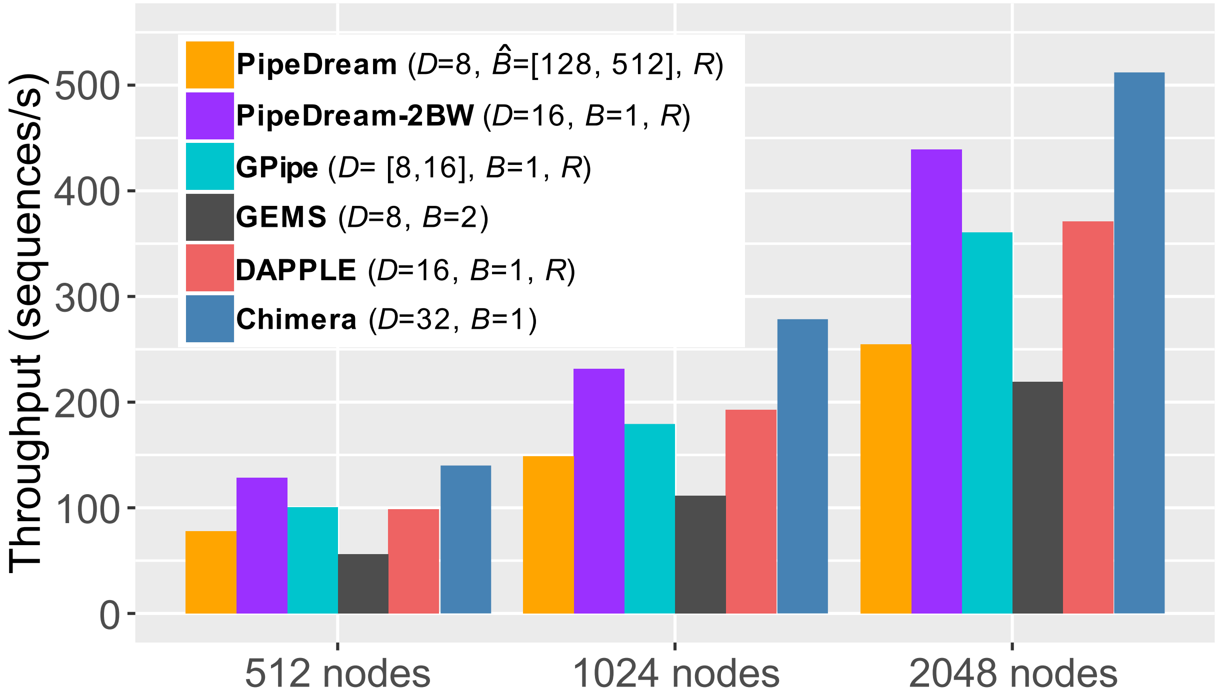



**Chimera** (backward is **2×** workload of forward)

S. Li, T. Hoefler: Chimera: Efficiently Training Large-Scale Neural Networks with Bidirectional Pipelines, best paper candidate at Supercomputing, SC21

# Chimera Weak Scaling (arXiv: 2107.06925v3)

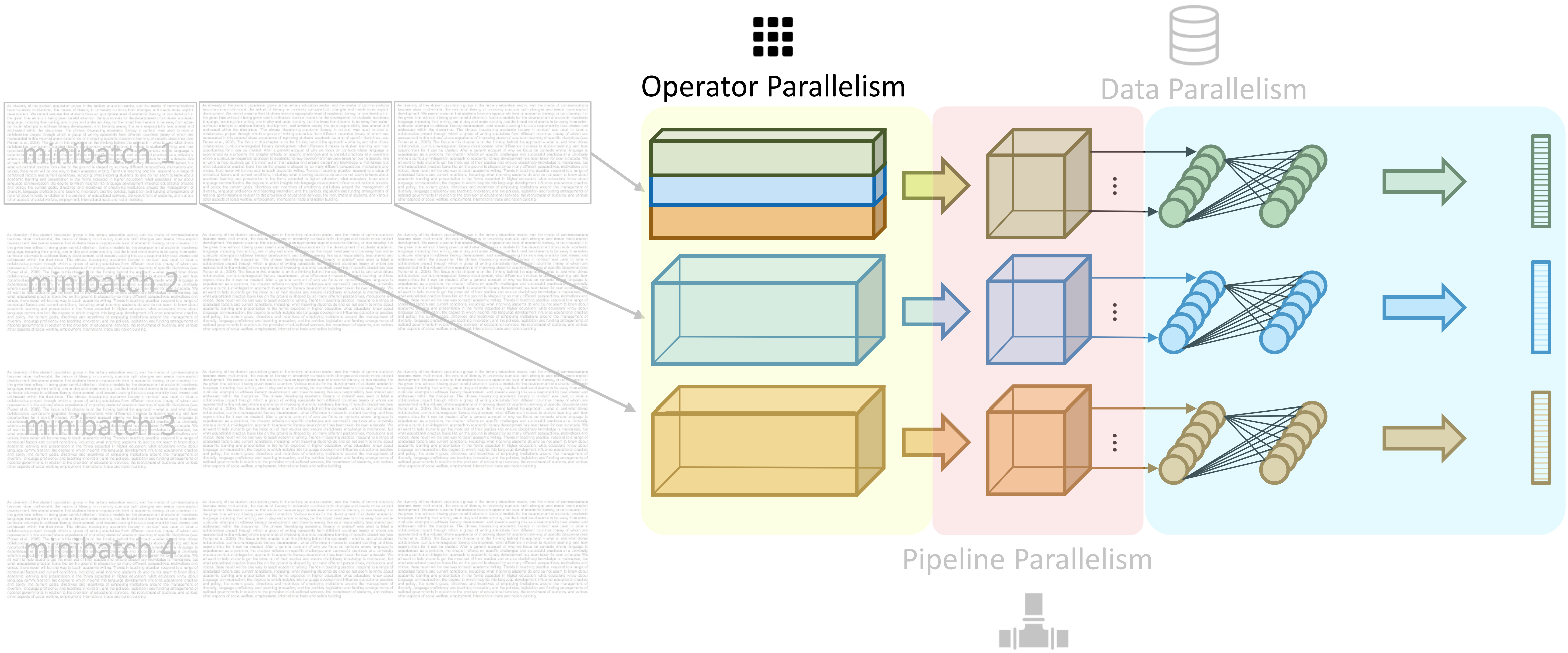



Weak scaling for GPT-2 on Piz Daint
(512 to 2048 GPU nodes)

- **1.38x - 2.34x speedup over synchronous approaches (GPipe, GEMS, DAPPLE)**
  - Less bubbles
  - More balanced memory thus no recomputation
- **1.16x - 2.01x speedup over asynchronous approaches (PipeDream-2BW, PipeDream)**
  - More balanced memory thus no recomputation
  - Gradient accumulation thus low synch frequency

S. Li, T. Hoefler: Chimera: Efficiently Training Large-Scale Neural Networks with Bidirectional Pipelines, best paper candidate at Supercomputing, SC21

# The Three Dimensions of Parallelism in Deep Learning (arXiv:1802.09941)



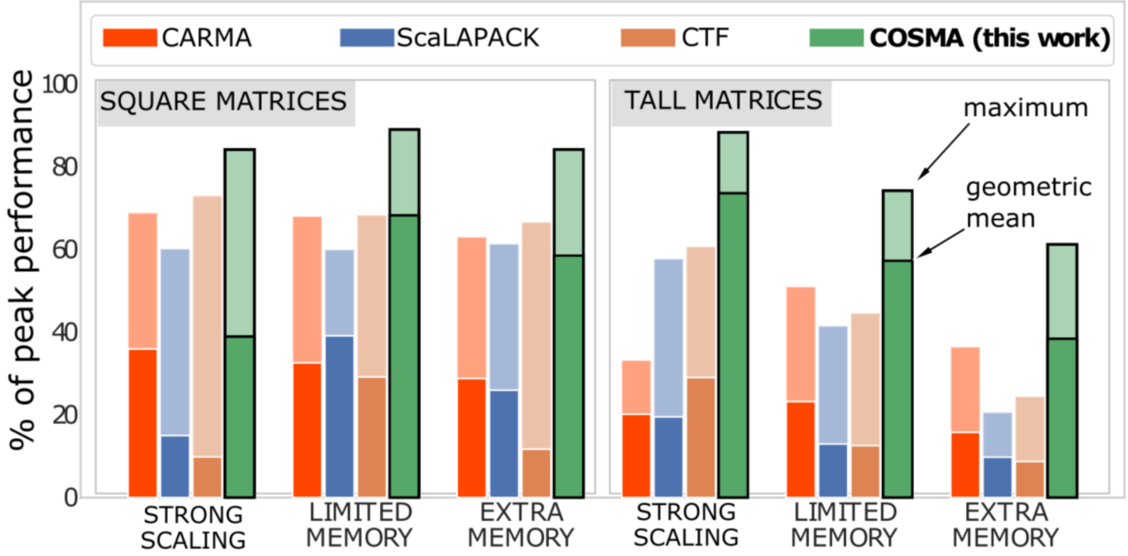

T. Ben-Nun, T. Hoefler: Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis, arXiv Feb 2018

# Operator Parallelism, i.e., Parallel Matrix Matrix Multiplication

Remember those?
All MMM!

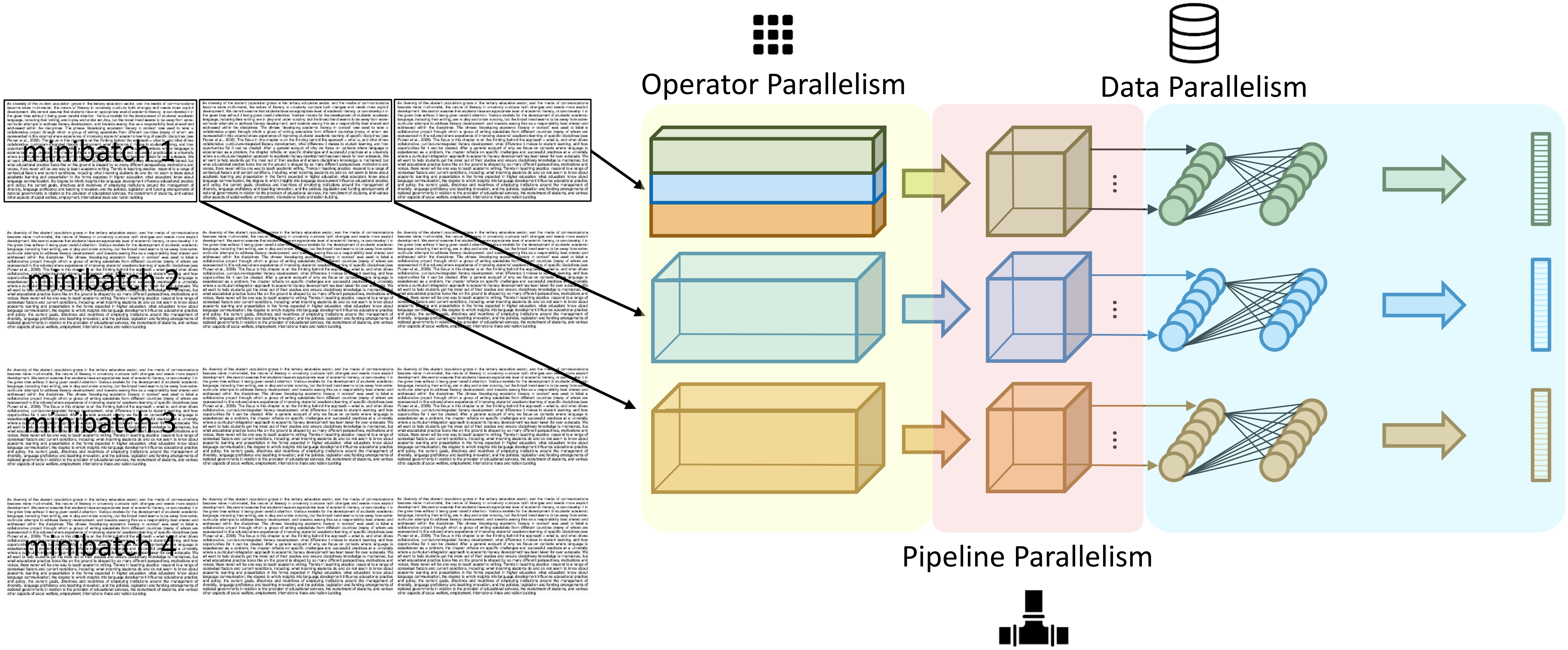
| Operator class | % flop | % Runtime |
|---|---|---|
| Tensor contraction | 99.80 | 61.0 |
| Statistical normalization | 0.17 | 25.5 |
| Element-wise | 0.03 | 13.5 |

- **Large MMMs dominate large language models!**
  - e.g., GPT-3 multiples 12,288x12,288 matrices
    *600 MiB in fp32 and 1.9 Tflop*
  - generative inference multiplies tall & skinny matrices
- **Distribute as operator parallelism**
  - Heaviest communication dimension!
    *Requires most optimization!*
- **COSMA [1] communication-optimal distributed MMM**
  - Achieves tight I/O lower bound of $Q \geq \min\left\{\frac{2mnk}{p\sqrt{S}} + S, 3\left(\frac{mnk}{p}\right)^{\frac{2}{3}}\right\}$
  - Uses partial replication with an outer-product schedule
    *See paper for details and proofs!*
- **AutoDDL [2] combines operator-parallel models into communication-avoiding data distribution**




[1] G. Kwasniewski et al.: "Red-Blue Pebbling Revisited: Near Optimal Parallel Matrix-Matrix Multiplication", best student paper at Supercomputing SC19
[2] J. Chen et al.: "AutoDDL: Automatic Distributed Deep Learning with Asymptotically Optimal Communication", arXiv

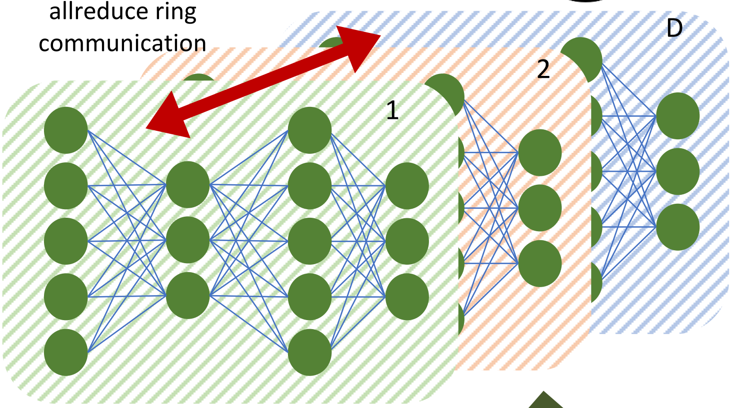
# The Three Dimensions of Parallelism in Deep Learning (arXiv:1802.09941)



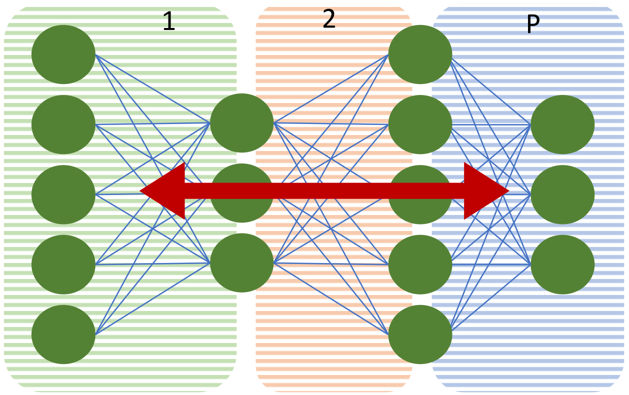

T. Ben-Nun, T. Hoefler: Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis, arXiv Feb 2018

# Communications in 3D Parallelism in Deep Learning (arXiv:2209.01346)

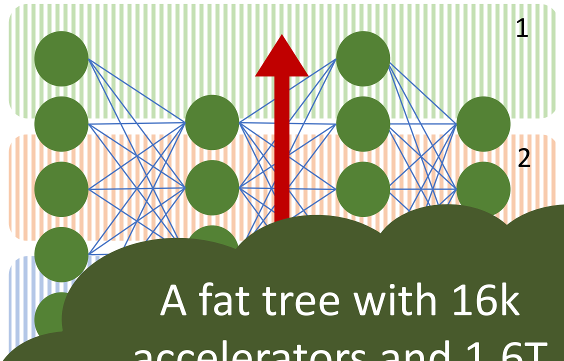
Data Parallelism




Pipeline Parallelism



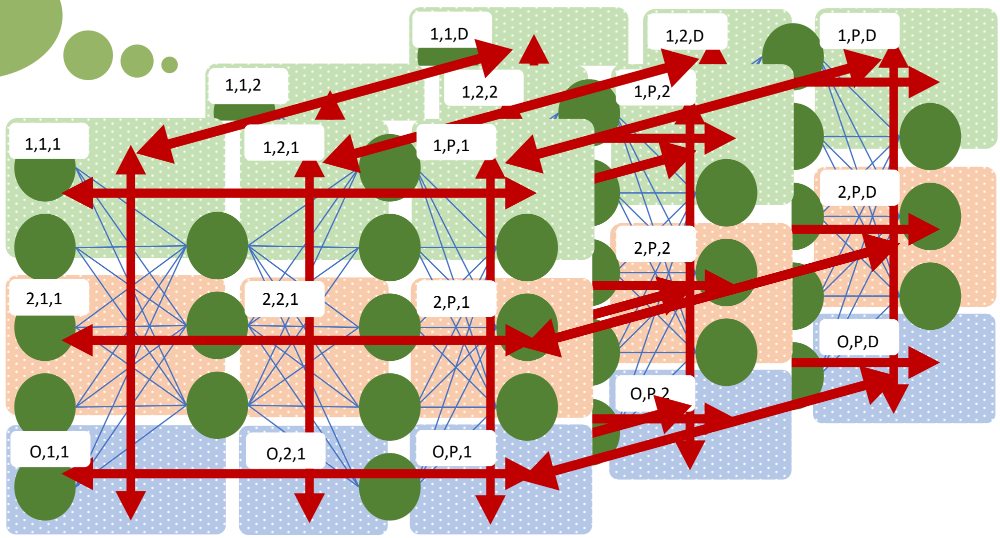

Operator Parallelism




A fat tree with 16k accelerators and 1.6T would cost $680M! ☹ ☹ ☹

Communication is (largely) a logical 3D Torus

3D - Data, Pipeline, and Operator Parallelism



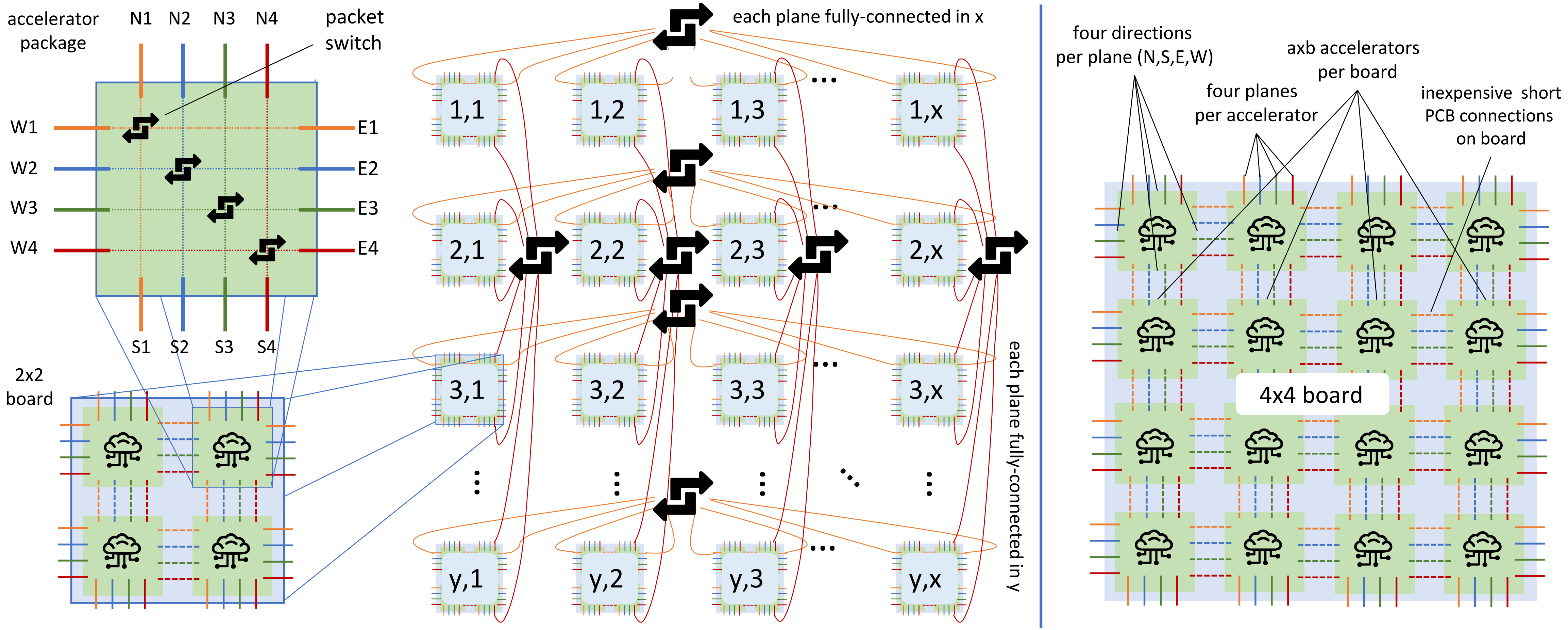

**AI bandwidth today / yesterday (and growing!)**

- Google TPUv2 (‘21): 1T
- AWS Trainium (‘21): 1.6T
- DGX-2 (A100, ‘21): 4.8T (islands of NVLINK)
- Tesla Dojo (‘22): 128T

→ Broadcom TH5 / NVIDIA Spectrum 4: 51.2T

TH et. al.: HammingMesh: A Network Topology for Large-Scale Deep Learning, to appear at SC22 and arXiv (2209.01346)

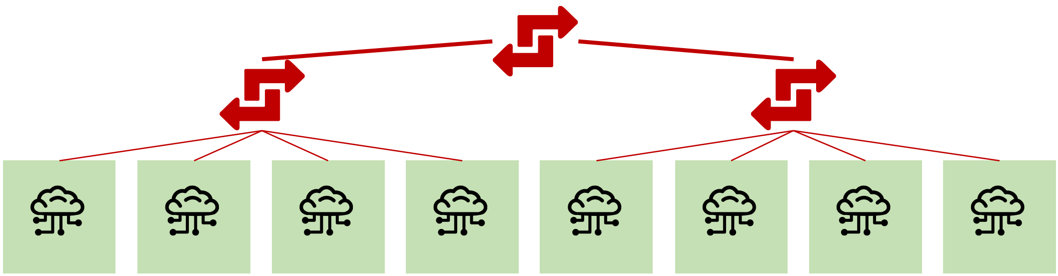
# Co-designing an AI Supercomputer with Unprecedented and Cheap Bandwidth



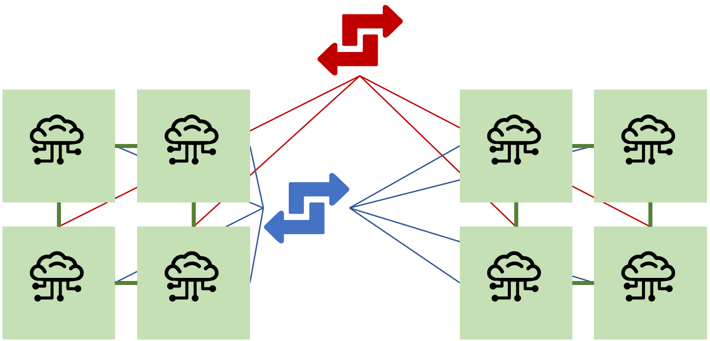

TH et. al.: HammingMesh: A Network Topology for Large-Scale Deep Learning, to appear at SC22 and arXiv (2209.01346)

# Bandwidth-cost-flexibility Tradeoffs (arXiv:2209.01346)

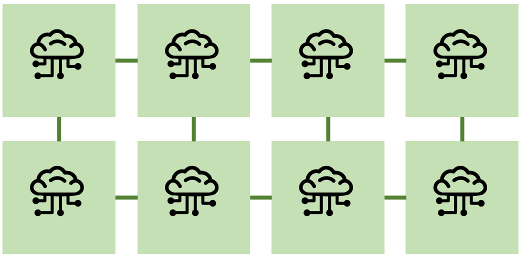
Global Topology
(e.g., Fat Tree)



**HammingMesh**
(many configurations)



Local Topology
(e.g., 2D Torus)

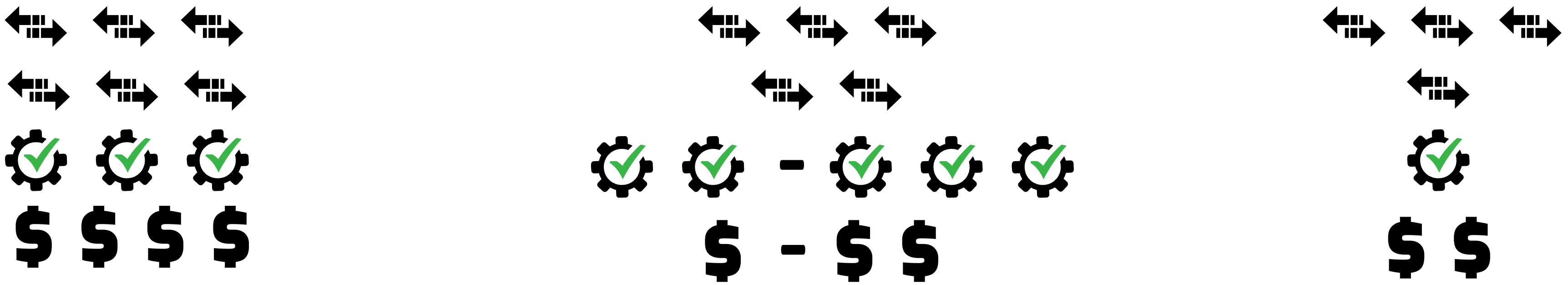


(large) reduce bandwidth

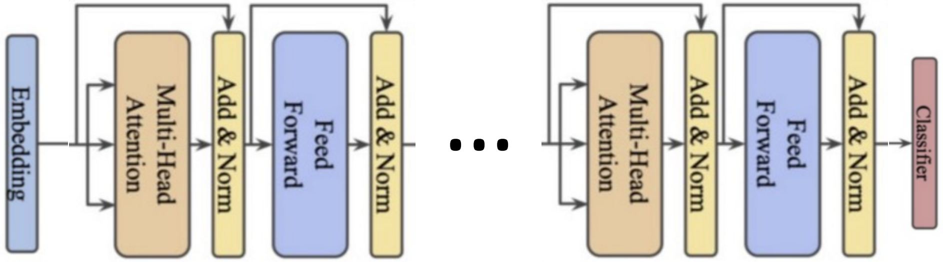
global bandwidth

placement flexibility

injection bandwidth



# Three Systems Dimensions in Large-scale Super-learning …




## High-Performance I/O

- Quickly **growing data volumes**
  - Scientific computing!
- Use the specifics of machine learning workloads
  - E.g., **intelligent prefetching**

## High-Performance Compute

- Deep learning is HPC
  - **Data movement!**
  - Specialization, Sparsification
  - Drives modern accelerators!

Data Movement Is All You Need: A Case Study on Optimizing Transformers

## High-Performance Communication

- Use larger clusters (10k+ GPUs)
- Model parallelism
  - Complex **pipeline schemes**
- Optimized networks

**Distribution and Parallelism**

Data | Pipeline | Operator

## What will the (near future bring)?

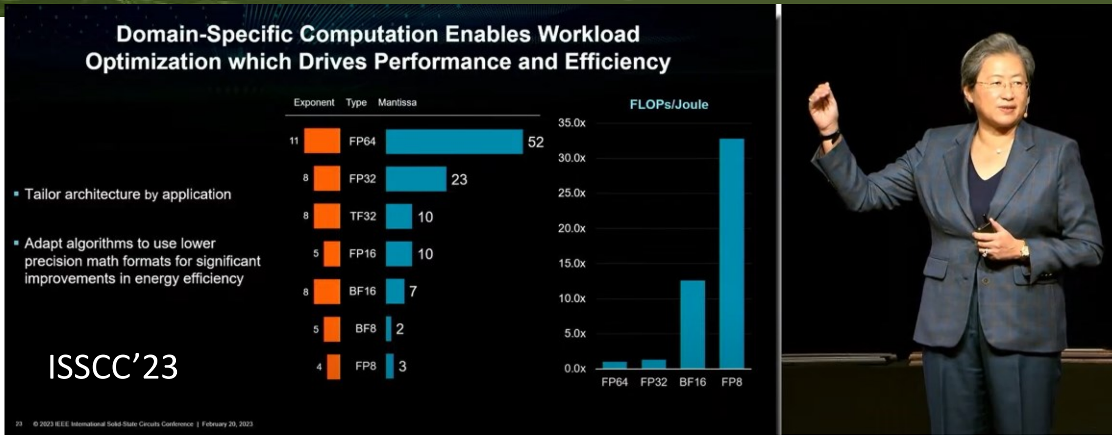
**Some predictions for the future of HPC but also computing at large!**

## Prediction 1: Accelerators Converge

# AI is a gravity well – HPC will follow

# Future Accelerators …

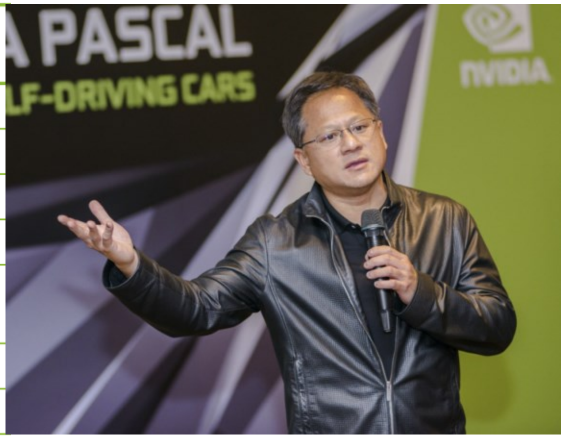
- **Most of the performance will be low precision arithmetic!**
  - I would predict (C)FP8 or smaller
  - We can be lucky if we get some fp64!
- **They will support quantization and sparsity in hardware**
  - Vector scaling and zero points
- **They will heavily be optimized towards data movement**
  - Physical limits and cost introduce two fundamental constraints:
    *Latency will become a problem*
    *Locality and sparse connectivity*
  - Potentially hard to program



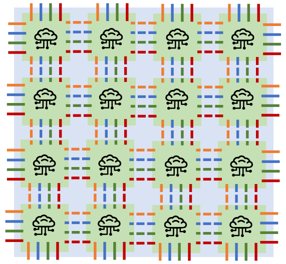

SPECIFICATIONS

| | H100 SXM |
|---|---|
| FP64 | 34 TFLOPS |
| FP64 Tensor Core | 67 TFLOPS |
| FP32 | 67 TFLOPS |
| TF32 Tensor Core | 989 TFLOPS* |
| BFLOAT16 Tensor Core | 1,979 TFLOPS* |
| FP16 Tensor Core | 1,979 TFLOPS* |
| FP8 Tensor Core | 3,958 TFLOPS* |
| INT8 Tensor Core | 3,958 TOPS* |

30x*




B. Wisniewski (Samsung)
**Memory-coupled Compute**
SPCL_Bcast 01/19/23
https://www.youtube.com/watch?v=KCrQtpx31CQ



Optimized topologies and network technologies.
E.g., HammingMesh
https://www.youtube.com/watch?v=xxwT45ljG4o

# Prediction 2: Programming and Tools Converge

# Data Science as a gravity well – HPC will follow

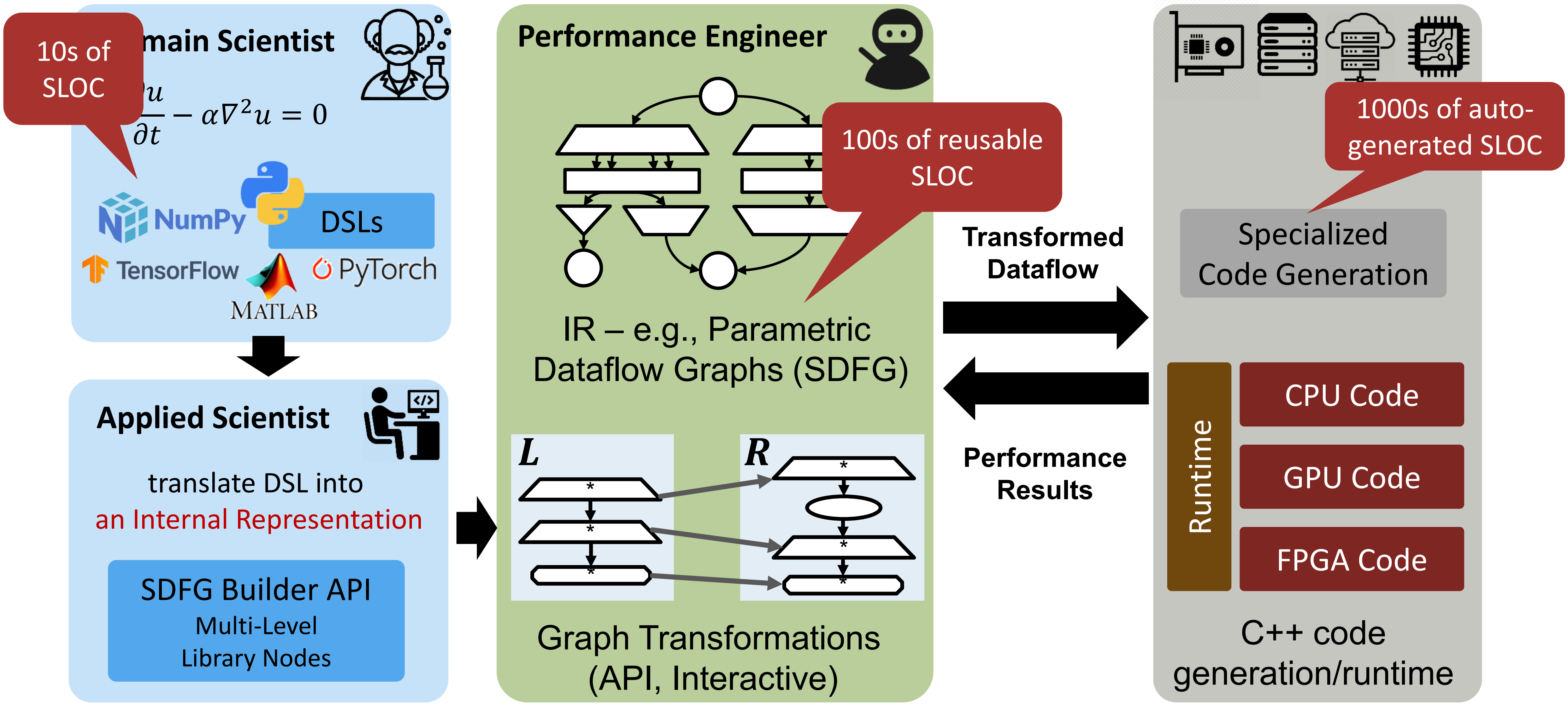
# Scientific Computing is Moving to Python (as language frontend/ecosystem)

Tiobe Index June'23

| Jun 2022 | Change | Programming Language |
|---|---|---|
| 1 | | Python |
| 2 | | C |
| 4 | ^ | C++ |
| 3 | v | Java |
| 5 | | C# |
| 6 | | Visual Basic |

439,100 projects

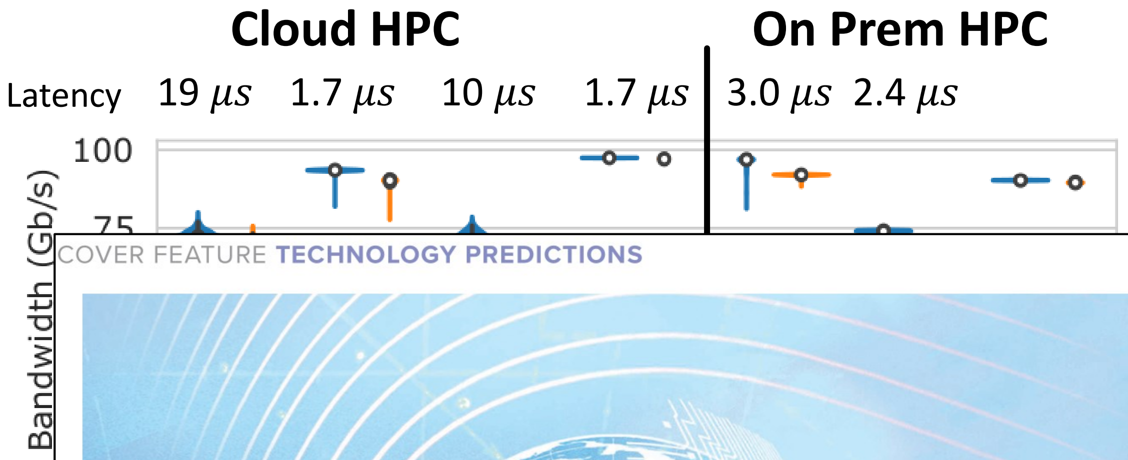
# Upleveling Programming in the 21st Century – Performance Metaprogramming




Ben-Nun, de Fine Licht, Ziogas, TH: Stateful Dataflow Multigraphs: A Data-Centric Model for High-Performance Parallel Programs, SC19

# Prediction 3: Networks Converge

# Cloud as a gravity well – HPC will follow




COVER FEATURE TECHNOLOGY PREDICTIONS

# The Convergence of Hyperscale Data Center and High-Performance Computing Networks

**Torsten Hoefler,** ETH Zurich
**Ariel Hendel,** Scala Computing
**Duncan Roweth,** Hewlett Packard Enterprise

*We discuss the differences and commonalities between network technologies used in supercomputers and data centers and outline a path to convergence at multiple layers. We predict that emerging smart networking solutions will accelerate that convergence.*

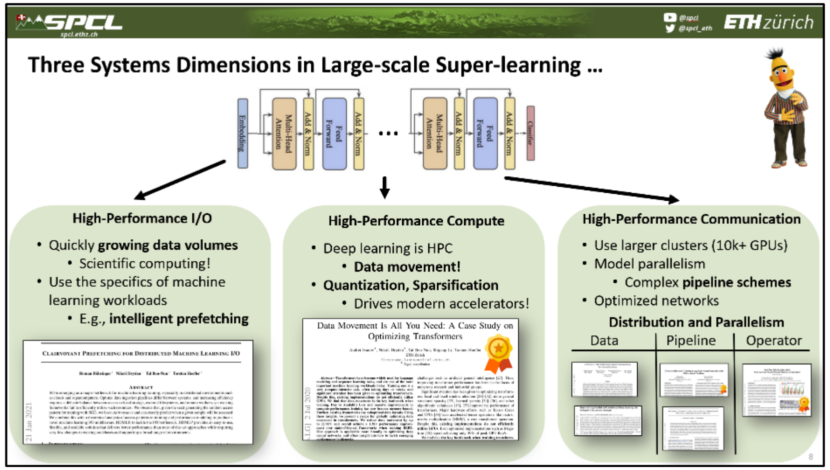
COVER FEATURE TECHNOLOGY PREDICTIONS

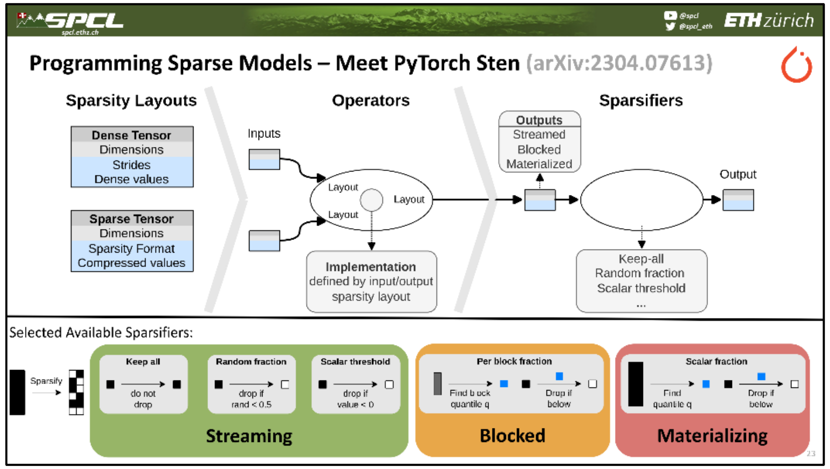
# Data Center Ethernet and Remote Direct Memory Access: Issues at Hyperscale

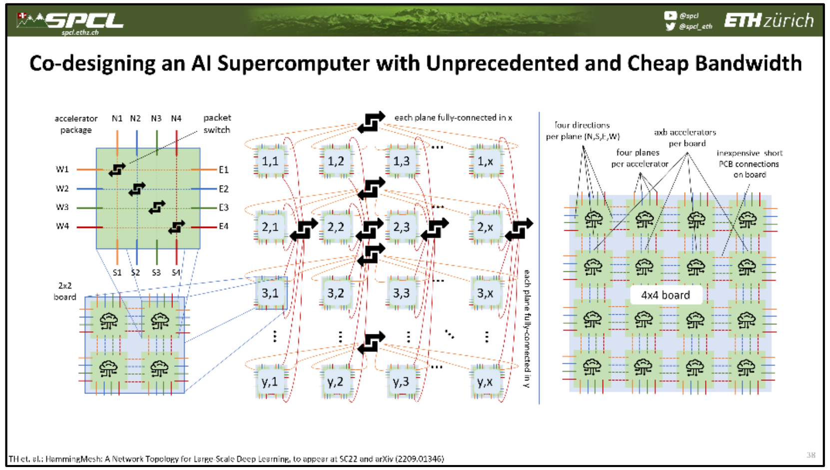
**Torsten Hoefler**, ETH Zürich
**Duncan Roweth, Keith Underwood, and Robert Alverson,** Hewlett Packard Enterprise
**Mark Griswold, Vahid Tabatabaee, Mohan Kalkunte, and Surendra Anubolu,** Broadcom
**Siyuan Shen,** ETH Zürich
**Moray McLaren,** Google
**Abdul Kabbani and Steve Scott,** Microsoft

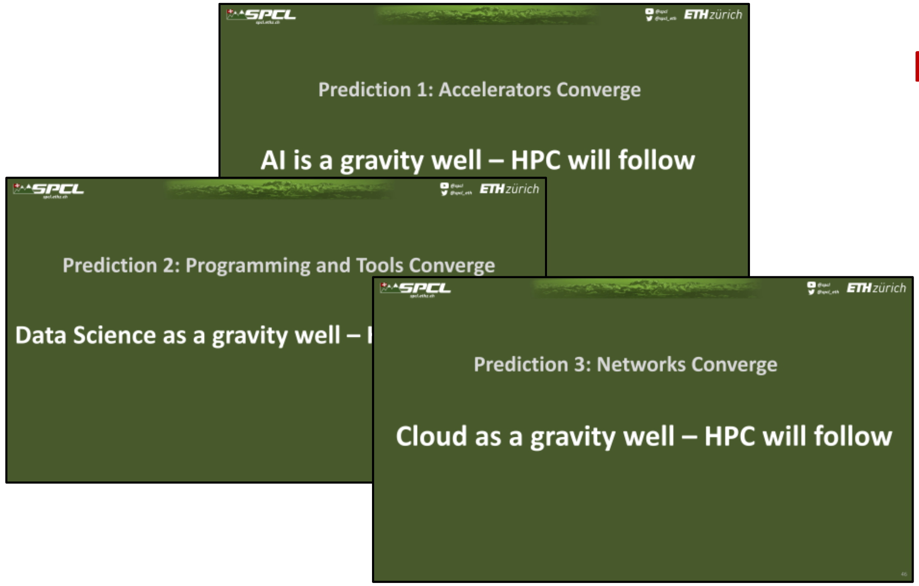
[1] De Sensi et al.: “Noise in the Clouds: Influence of Network Performance Variability on Application Scalability”, SIGMETRICS’23

# Key Points and Conclusions




## More of SPCL's research:

- youtube.com/@spcl — 180+ Talks
- twitter.com/spcl_eth — 1.4K+ Followers
- github.com/spcl — 2K+ Stars

… or spcl.ethz.ch

Want to join our efforts?
We're looking for excellent
Postdocs, PhD students, and Visitors.
Talk to me!