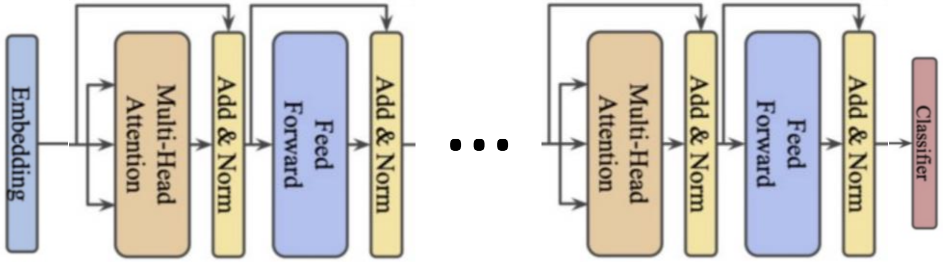
T. HOEFLER

The Three Pillars of Large-scale Deep Learning

with contributions by the whole SPCL deep learning team (T. Ben-Nun, S. Li, N. Dryden and many others) and collaborators (D. Alistarh and others)

Pushing the envelope in large-scale learning



High-Performance I/O

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

21 Jan 2021

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.

High-Performance Compute

- Deep learning is HPC
  - Same major problems
  - **Data movement!**

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

[cs.LG] 2 Jul 2020

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-**

High-Performance Communication

- Use larger clusters (10k+ GPUs)
- Model parallelism
  - Complex pipeline schemes
- Sparsification

SparCML: High-Performance Sparse Communication for Machine Learning

Cedric Renggli, ETH Zurich; Saleh Ashkboos, IST Austria; Mehdi Aghagolzadeh, Microsoft; Dan Alistarh, IST Austria
Torsten Hoefler, ETH Zurich

[cs.DC] 16 Aug 2019

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

TAL BEN-NUN and TORSTEN HOEFLER, ETH Zurich, Switzerland

15 Sep 2018

Deep Neural Networks (DNNs) are becoming an important tool in modern computing applications. Accelerating their training is a major challenge and techniques range from distributed algorithms to low-level circuit design. In this survey, we describe the problem from a theoretical perspective, followed by approaches for its parallelization. We present trends in DNN architectures and the resulting implications on parallelization strategies. We then review and model the different types of concurrency in DNNs: from the single operator, through parallelism in network inference and training, to distributed deep learning. We discuss asynchronous stochastic optimization, distributed system architectures, communication schemes, and neural architecture search. Based on those approaches, we extrapolate potential directions for parallelism in deep learning.

CCS Concepts: • **General and reference** → *Surveys and overviews*; • **Computing methodologies** → **Neural networks**; **Parallel computing methodologies**; **Distributed computing methodologies**;

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 on one A100 - 2.9k samples/s → 333 MiB/s random access → 2 SSDs / GPU

    2-4x that for scientific problem such as CosmoFlow
- **Training on thousands of GPUs may need to manage k × 1000s of SSDs**

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

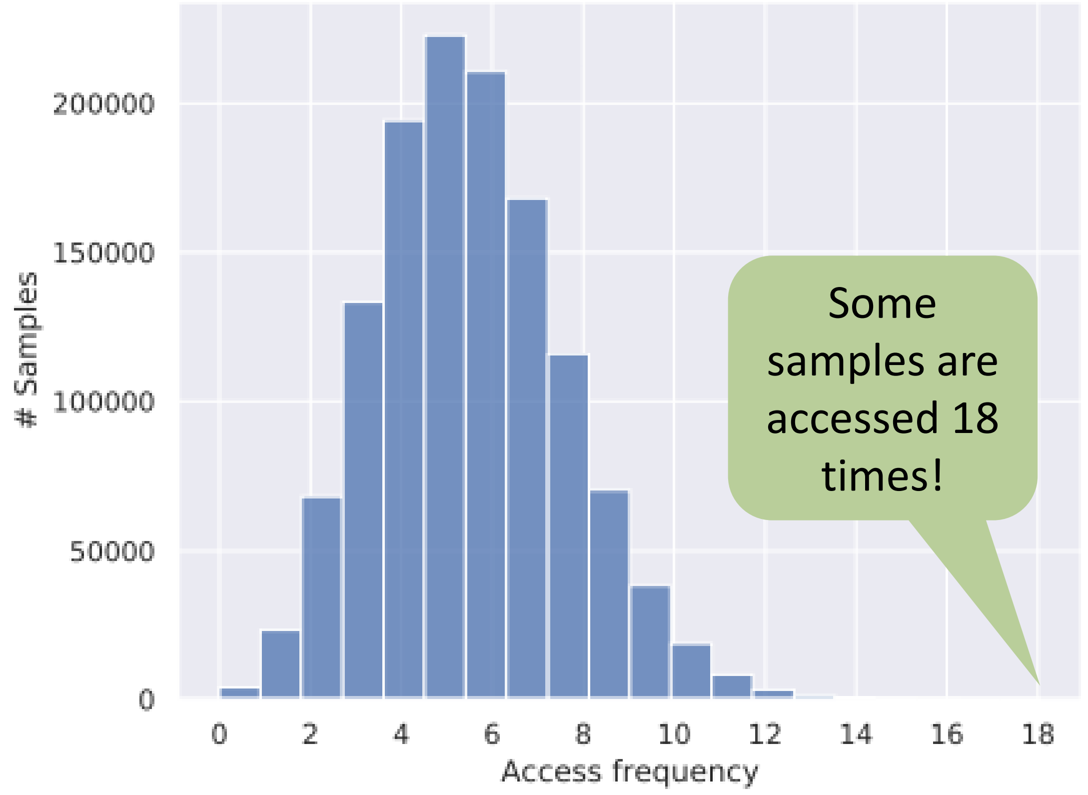
- **But why do we need those even? Deep Learning workloads "randomly sample" input!**
  - By "random", we really mean pseudo-random sequences with fixed seeds ☺

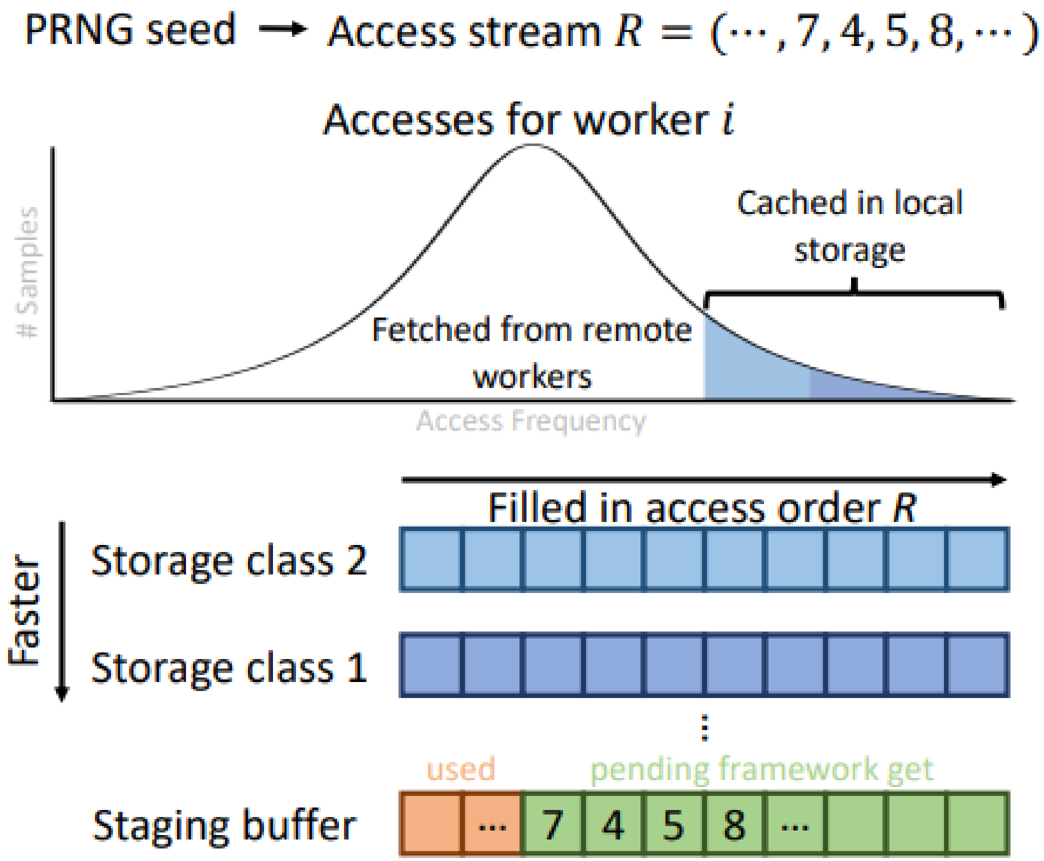
    This enables clairvoyant prefetching!

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**!

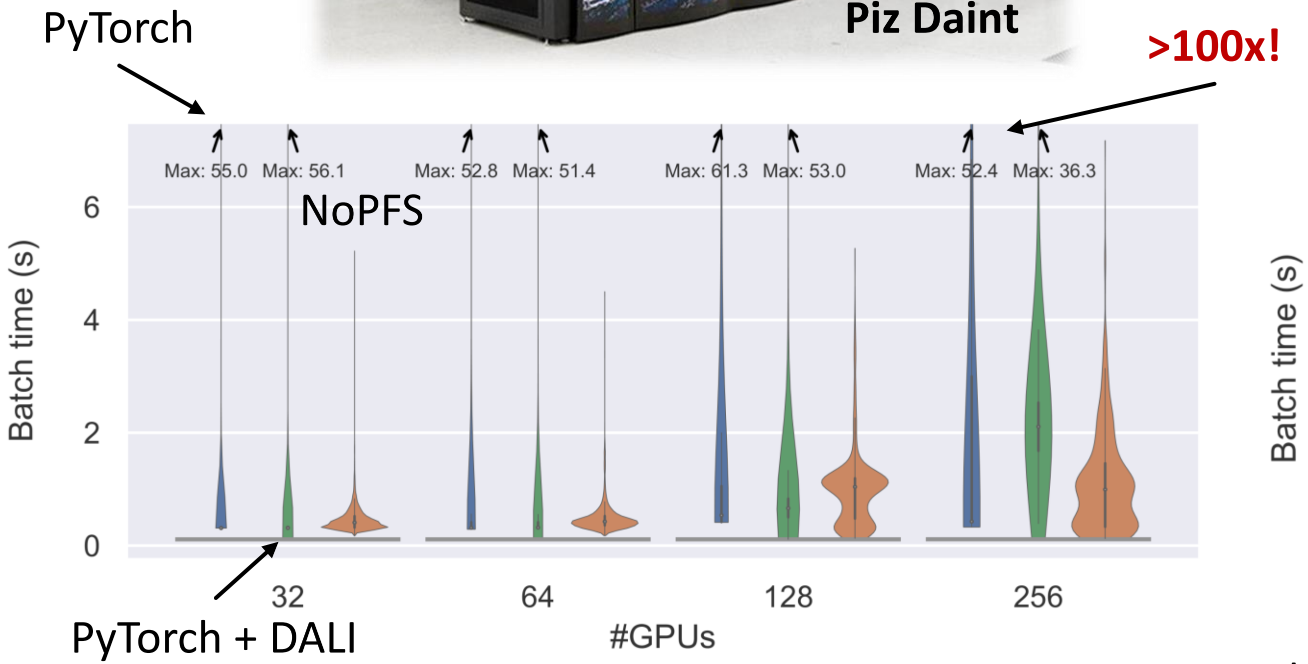
single-process access to samples
for ImagerNet with 16 processes

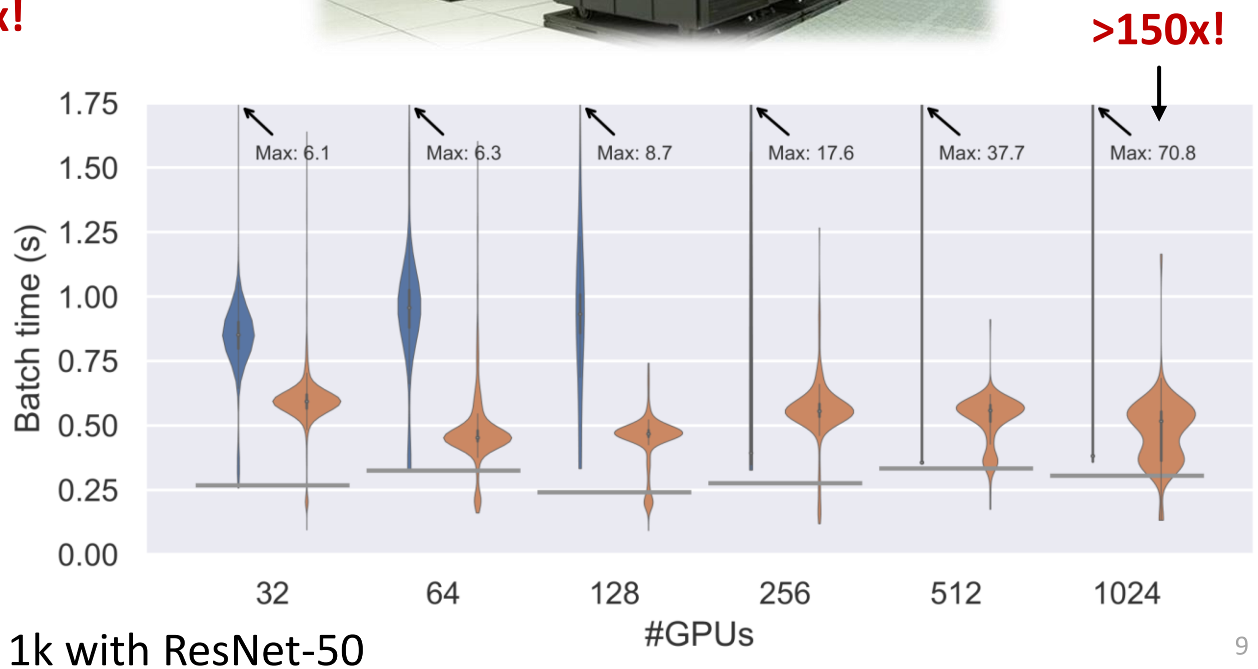




Clairvoyant Prefetching for Distributed Machine Learning I/O (arXiv 2101.08734)

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



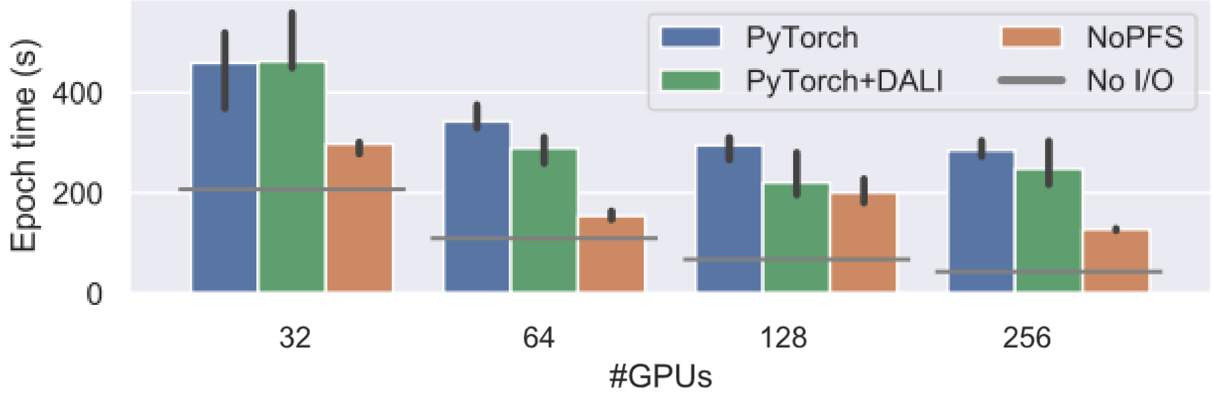


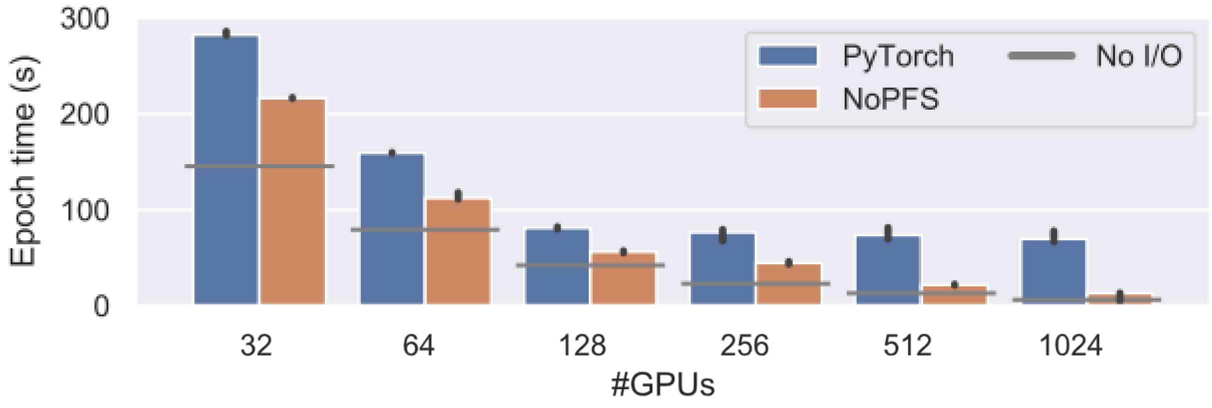
ImageNet 1k with ResNet-50

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!

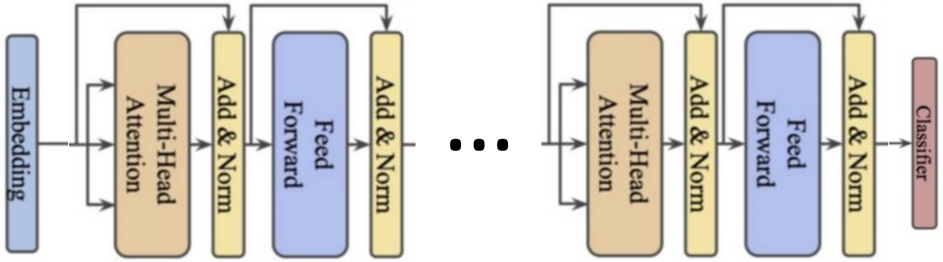
runtime per epoch (full training time)





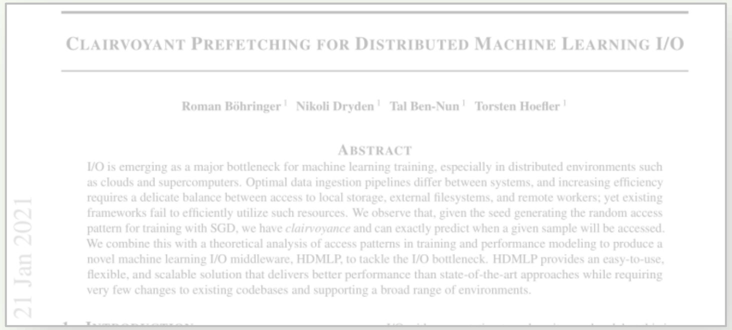
ImageNet 1k with ResNet-50

Pushing the envelope in large-scale 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
  - Same major problems
  - **Data movement!**

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.

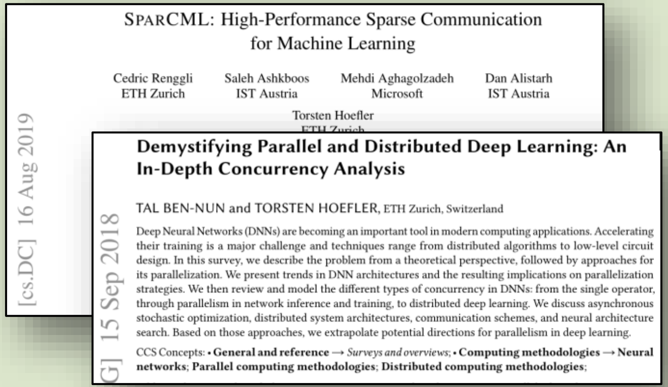
Significant attention has been given to optimizing transformers: local and fixed-window attention [28]–[32], more general structured sparsity [33], learned sparsity [34]–[36], and other algorithmic techniques [19], [37] improve the performance of transformers. Major hardware efforts, such as Tensor Cores 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
- Sparsification



SPARCML: High-Performance Sparse Communication for Machine Learning

Cedric Renggli, ETH Zurich; Saleh Ashkboos, IST Austria; Mehdi Aghagolzadeh, Microsoft; Dan Alistarh, IST Austria; Torsten Hoefler, ETH Zurich

[cs.DC] 16 Aug 2019

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

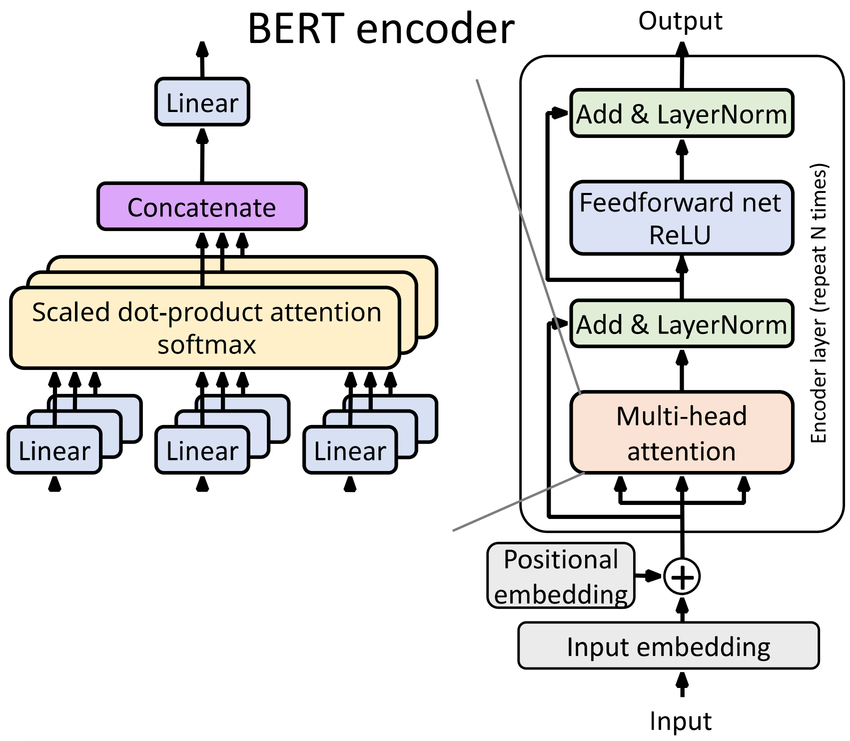
TAL BEN-NUN and TORSTEN HOEFLER, ETH Zurich, Switzerland

Deep Neural Networks (DNNs) are becoming an important tool in modern computing applications. Accelerating their training is a major challenge and techniques range from distributed algorithms to low-level circuit design. In this survey, we describe the problem from a theoretical perspective, followed by approaches for its parallelization. We present trends in DNN architectures and the resulting implications on parallelization strategies. We then review and model the different types of concurrency in DNNs: from the single operator, through parallelism in network inference and training, to distributed deep learning. We discuss asynchronous stochastic optimization, distributed system architectures, communication schemes, and neural architecture search. Based on those approaches, we extrapolate potential directions for parallelism in deep learning.

CCS Concepts: • **General and reference** → *Surveys and overviews*; • **Computing methodologies** → **Neural networks**; **Parallel computing methodologies**; **Distributed computing methodologies**;

G] 15 Sep 2018

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%

Our performance improvement for BERT-large

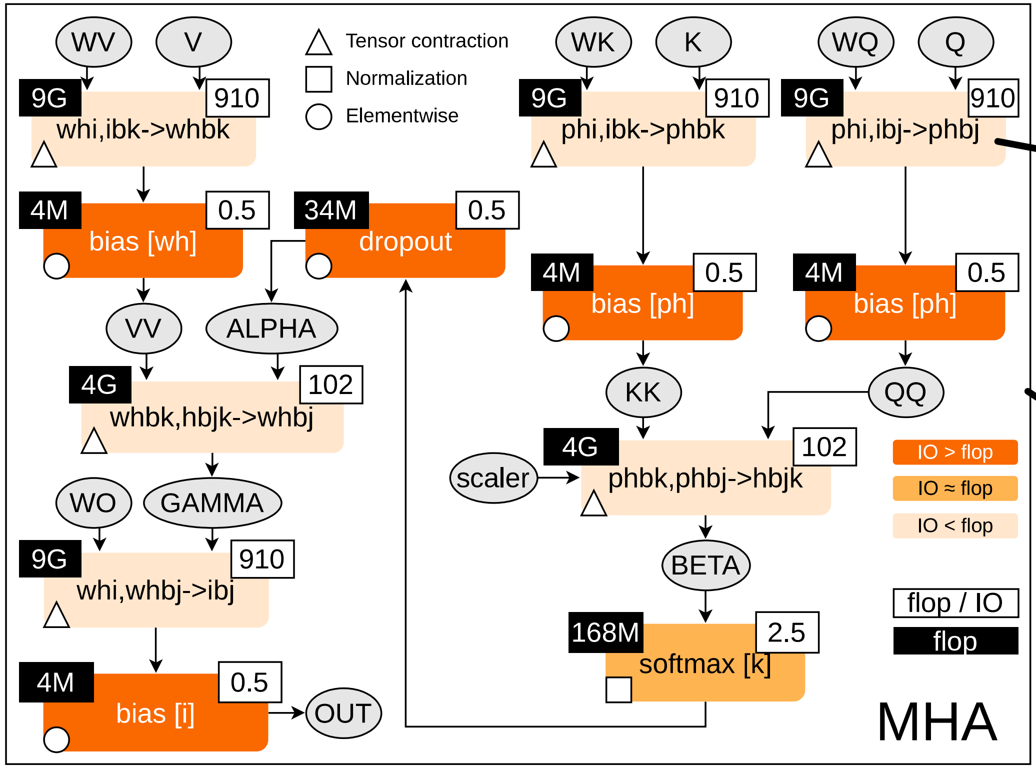
- 30% over PyTorch
- 20% over Tensorflow + XLA
- 8% over DeepSpeed

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

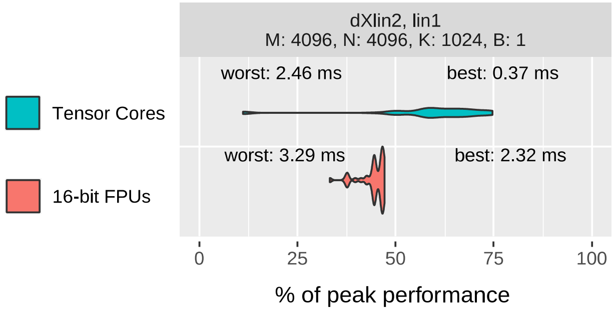
Last week, OpenAI published a paper detailing GPT-3, a machine learning model that achieves strong results on a number of natural language benchmarks. At 175 billion parameters, where a parameter affects data's prominence in an overall prediction, it's the largest of its kind. And with a memory size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

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)

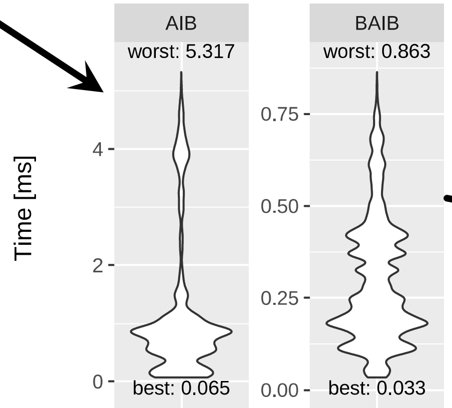


different data layouts

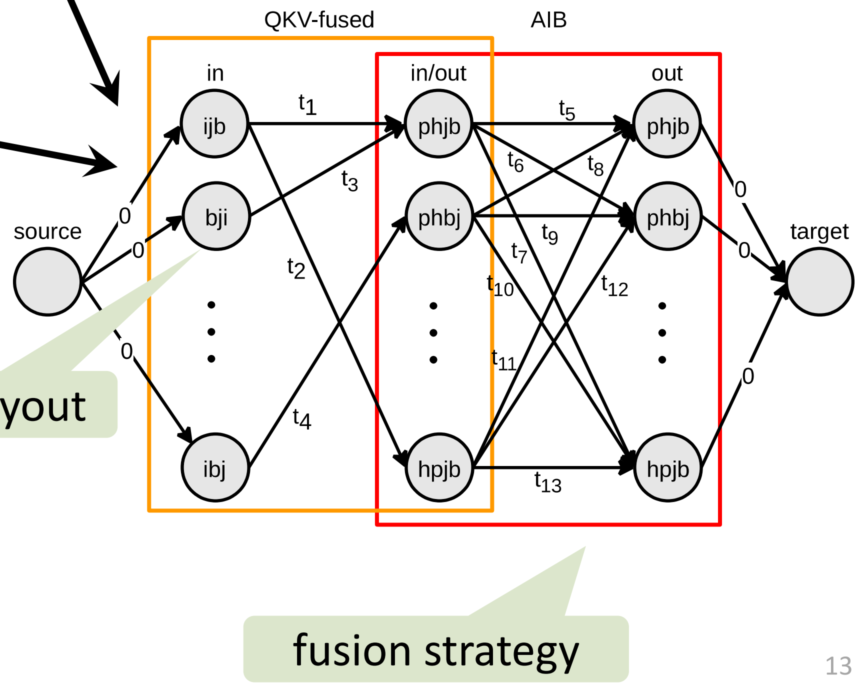


https://github.com/spcl/dace

different fusion strategies



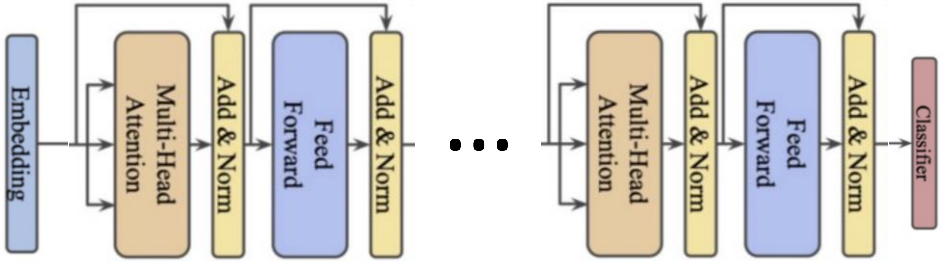
Configuration selection graph



Full BERT encoder layer performance (ms)

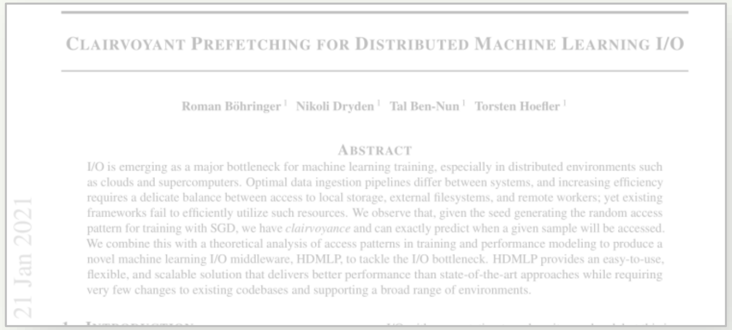
	TF+XLA	PyTorch	DeepSpeed	Ours
Forward	3.2	3.45	2.8	**2.63**
Backward	5.2	5.69	4.8	**4.38**

Pushing the envelope in large-scale 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.

High-Performance Compute

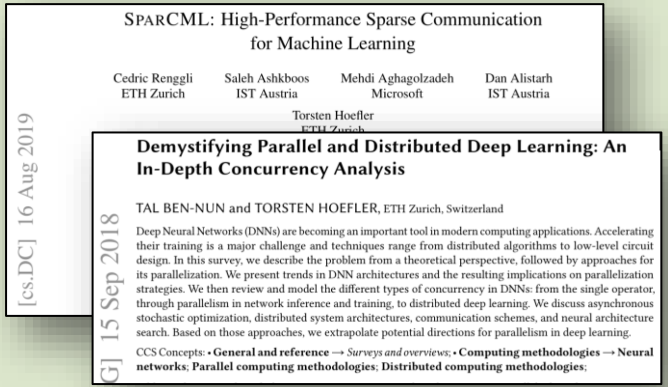
- Deep learning is HPC
  - Same major problems
  - **Data movement!**

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

High-Performance Communication

- Use larger clusters (10k+ GPUs)
- Model parallelism
  - Complex pipeline schemes
- Sparsification



SPARCML: High-Performance Sparse Communication for Machine Learning

Cedric Renggli, ETH Zurich; Saleh Ashkboos, IST Austria; Mehdi Aghagolzadeh, Microsoft; Dan Alistarh, IST Austria; Torsten Hoefler, ETH Zurich

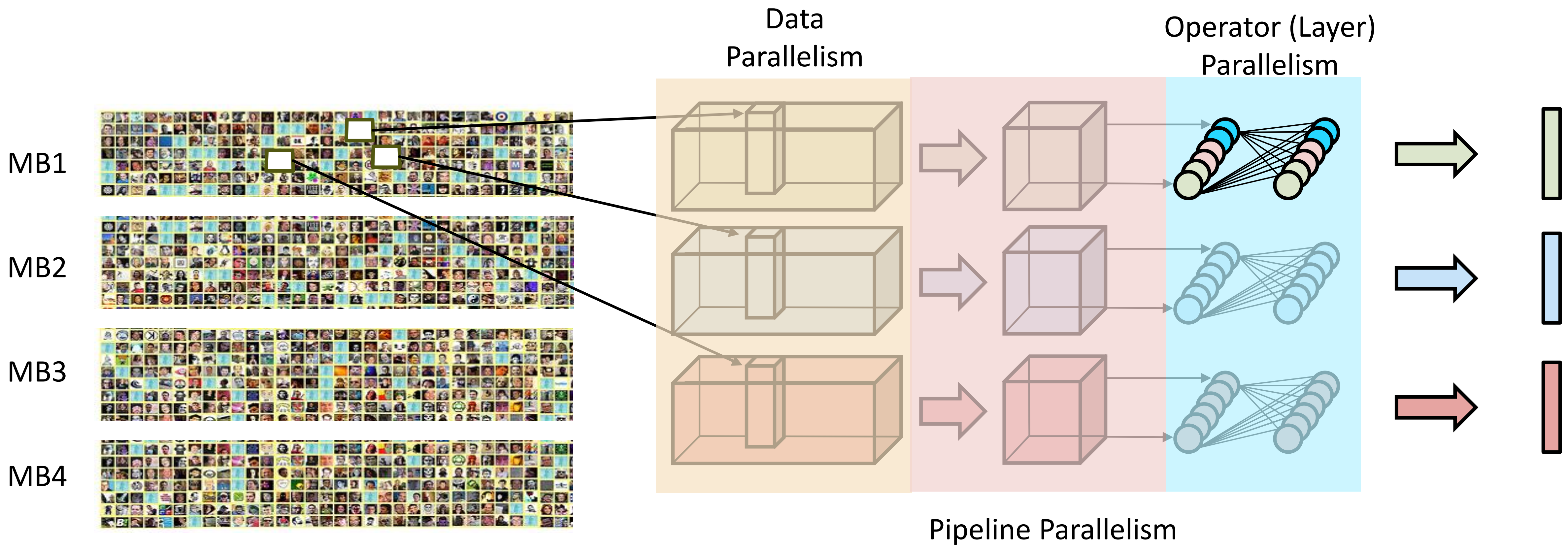
Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis

TAL BEN-NUN and TORSTEN HOEFLER, ETH Zurich, Switzerland

Deep Neural Networks (DNNs) are becoming an important tool in modern computing applications. Accelerating their training is a major challenge and techniques range from distributed algorithms to low-level circuit design. In this survey, we describe the problem from a theoretical perspective, followed by approaches for its parallelization. We present trends in DNN architectures and the resulting implications on parallelization strategies. We then review and model the different types of concurrency in DNNs: from the single operator, through parallelism in network inference and training, to distributed deep learning. We discuss asynchronous stochastic optimization, distributed system architectures, communication schemes, and neural architecture search. Based on those approaches, we extrapolate potential directions for parallelism in deep learning.

CCS Concepts: • General and reference → Surveys and overviews; • Computing methodologies → Neural networks; Parallel computing methodologies; Distributed computing methodologies;

The three dimensions of parallelism in deep learning (arXiv:1802.09941)



- **Large-scale deep learning will need all three dimensions!**
  - Depends on the exact model configuration
  - Sparsity makes it much more complex (interesting, more later)!

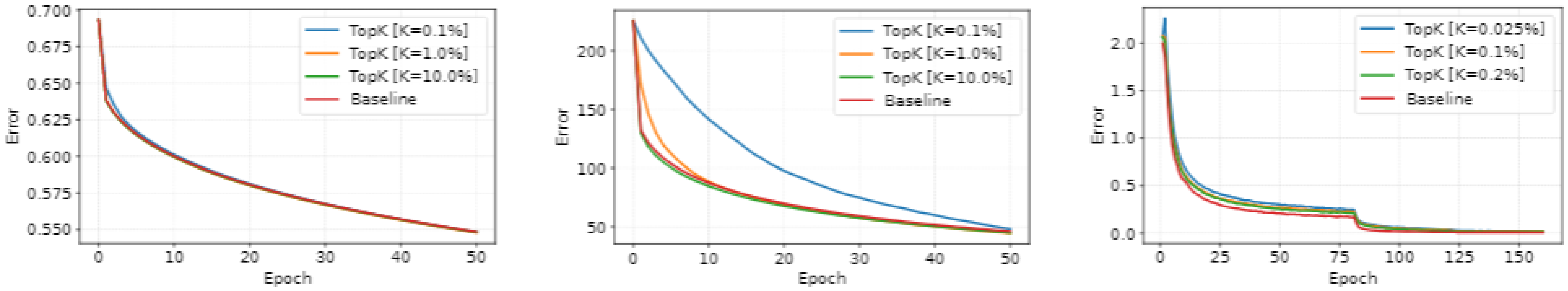
Data-parallel gradient sparsification – top-k SGD (arXiv:1809.10505)

- **Turns out 90-99.9% of the gradient values can be skipped by choosing the top-k – achieve similar accuracy**
  - Accumulate the remainder locally (convergence proof, similar to async. SGD with implicit staleness bounds [1])

Assumption 1. *There exists a (small) constant ξ such that, for every iteration $t \geq 0$, we have:*

$$\left\| \mathrm{TopK}\left(\frac{1}{P}\sum_{p=1}^{P}\left(\alpha\tilde{G}_t^p(v_t)+\epsilon_t^p\right)\right) - \sum_{p=1}^{P}\frac{1}{P}\mathrm{TopK}\left(\alpha\tilde{G}_t^p(v_t)+\epsilon_t^p\right)\right\| \leq \xi\|\alpha\tilde{G}_t(v_t)\|.$$

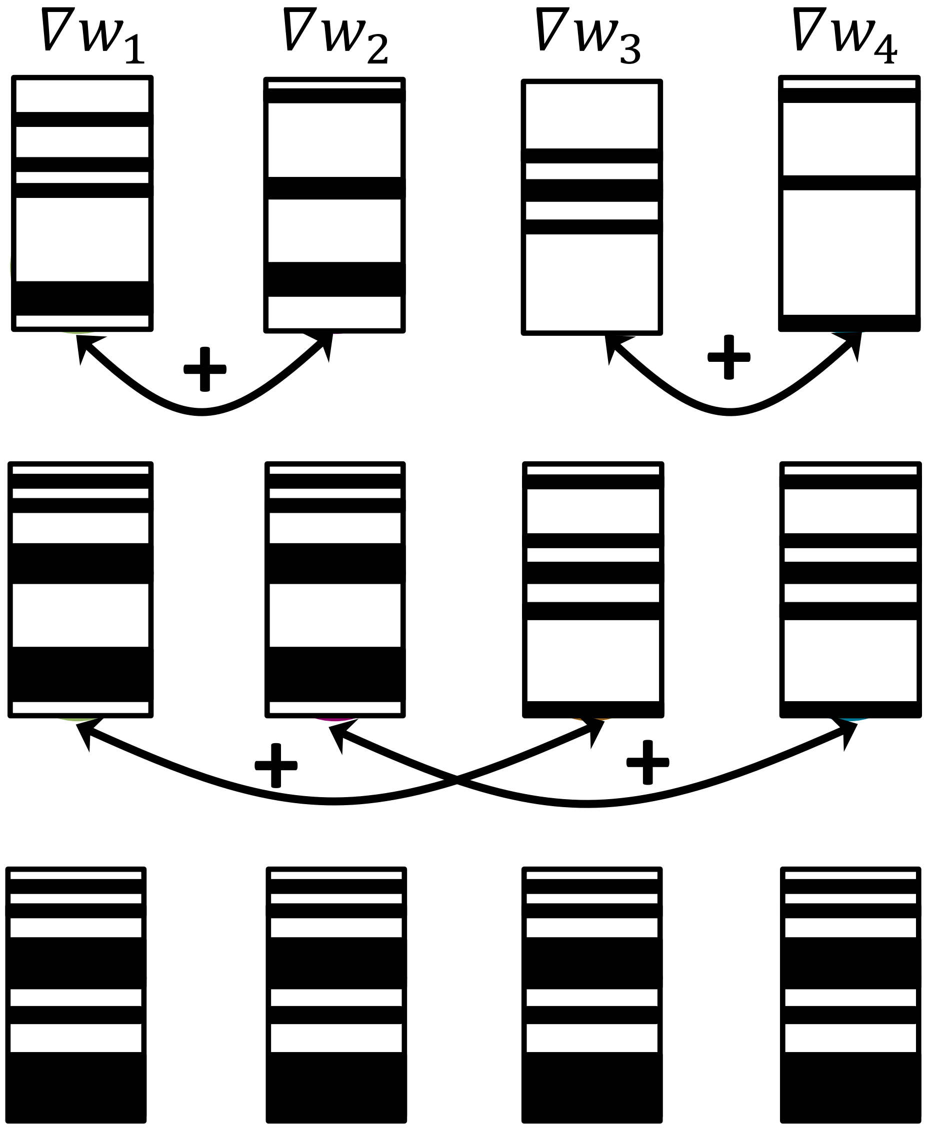
Discussion. We validate Assumption 1 experimentally on a number of different learning tasks in Section 6 (see also Figure 1). In addition, we emphasize the following points:



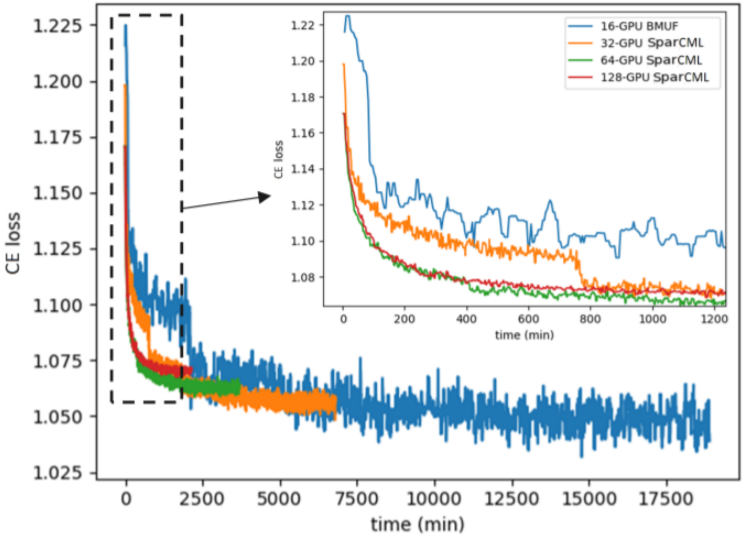
(a) RCV1 convergence. (b) Linear regression. (c) ResNet110 on CIFAR10.

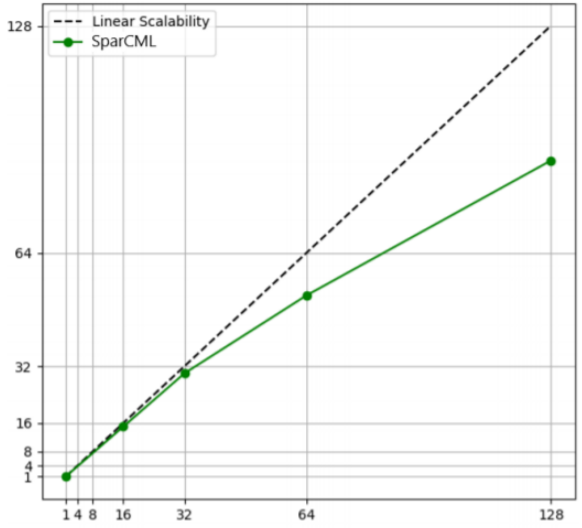
[1] Dan Alistarh, TH, et al.: “The Convergence of Sparsified Gradient Methods”, NIPS’18

SparCML – Sparse allreduce for decentral updates (arXiv:1802.08021)



Six epochs, 60 million params





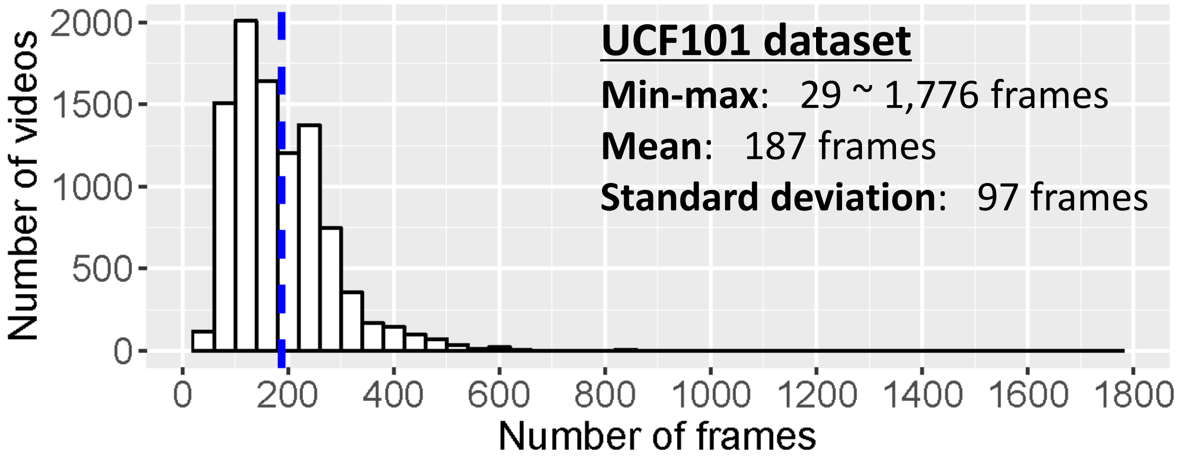
Microsoft Speech Production Workload Results – **2 weeks → 2 days**!

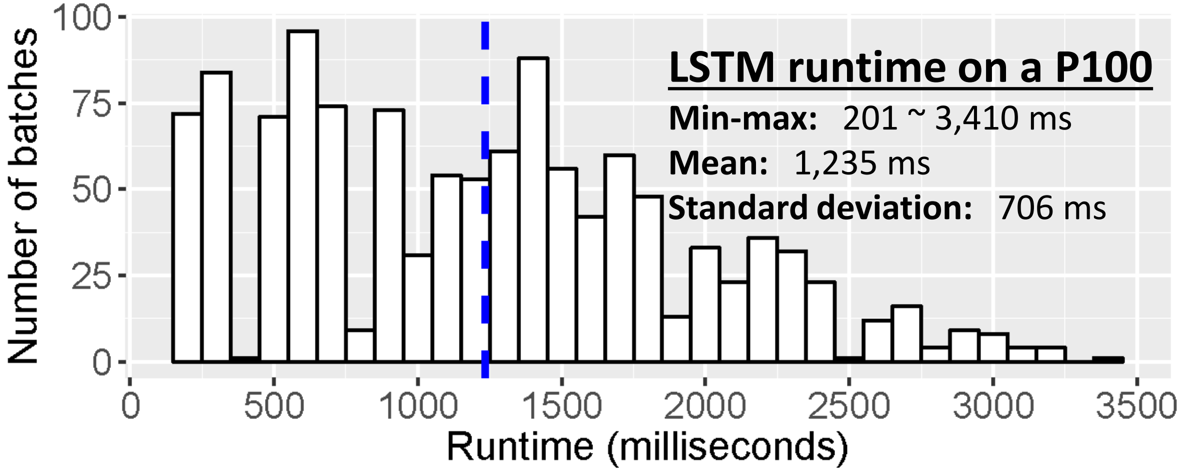
System	Dataset	Model	# of nodes	Algorithm	Speedup
Piz Daint	ImageNet	VGG19	8	Q4	1.55 (3.31)
Piz Daint	ImageNet	AlexNet	16	Q4	1.30 (1.36)
Piz Daint EC2	MNIST	MLP	8	Top16_Q4 Top16_Q4	3.65 (4.53) 19.12 (22.97)

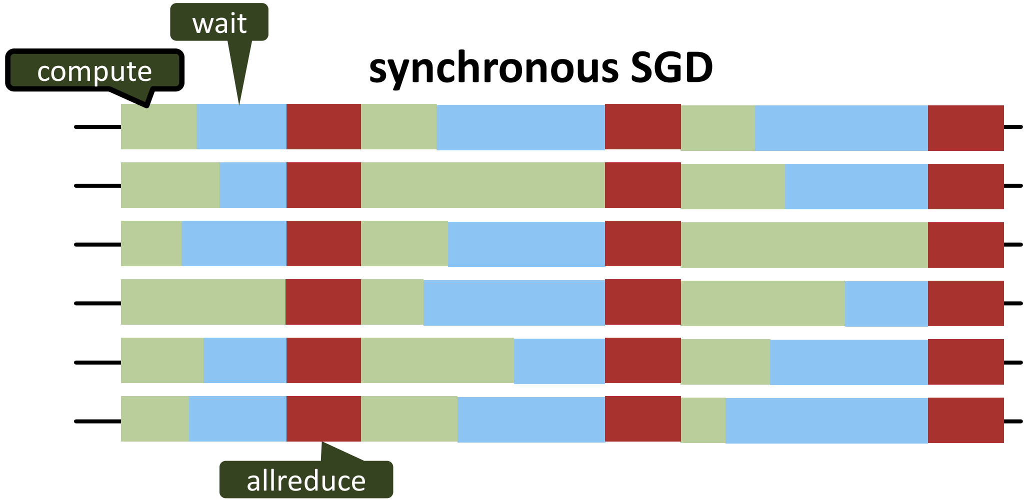
C. Renggli, TH et al. SparCML: High-Performance Sparse Communication for Machine Learning, SC19

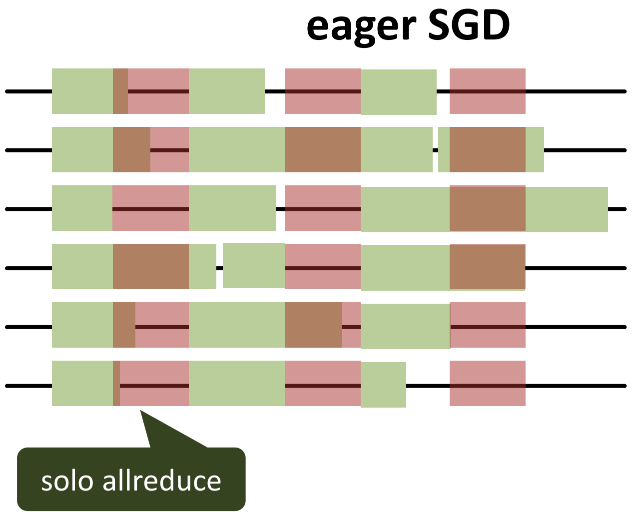
Unbalanced workloads in deep learning → eager-SGD (arXiv:1908.04207)

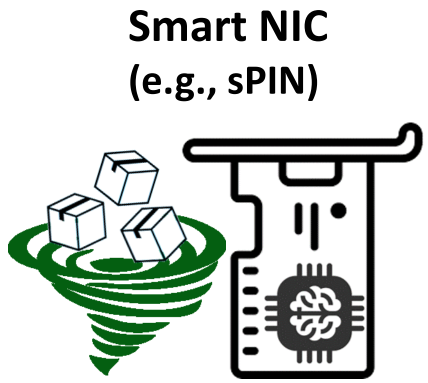
Some training scenarios cause different work across examples – e.g., video inputs of different lengths



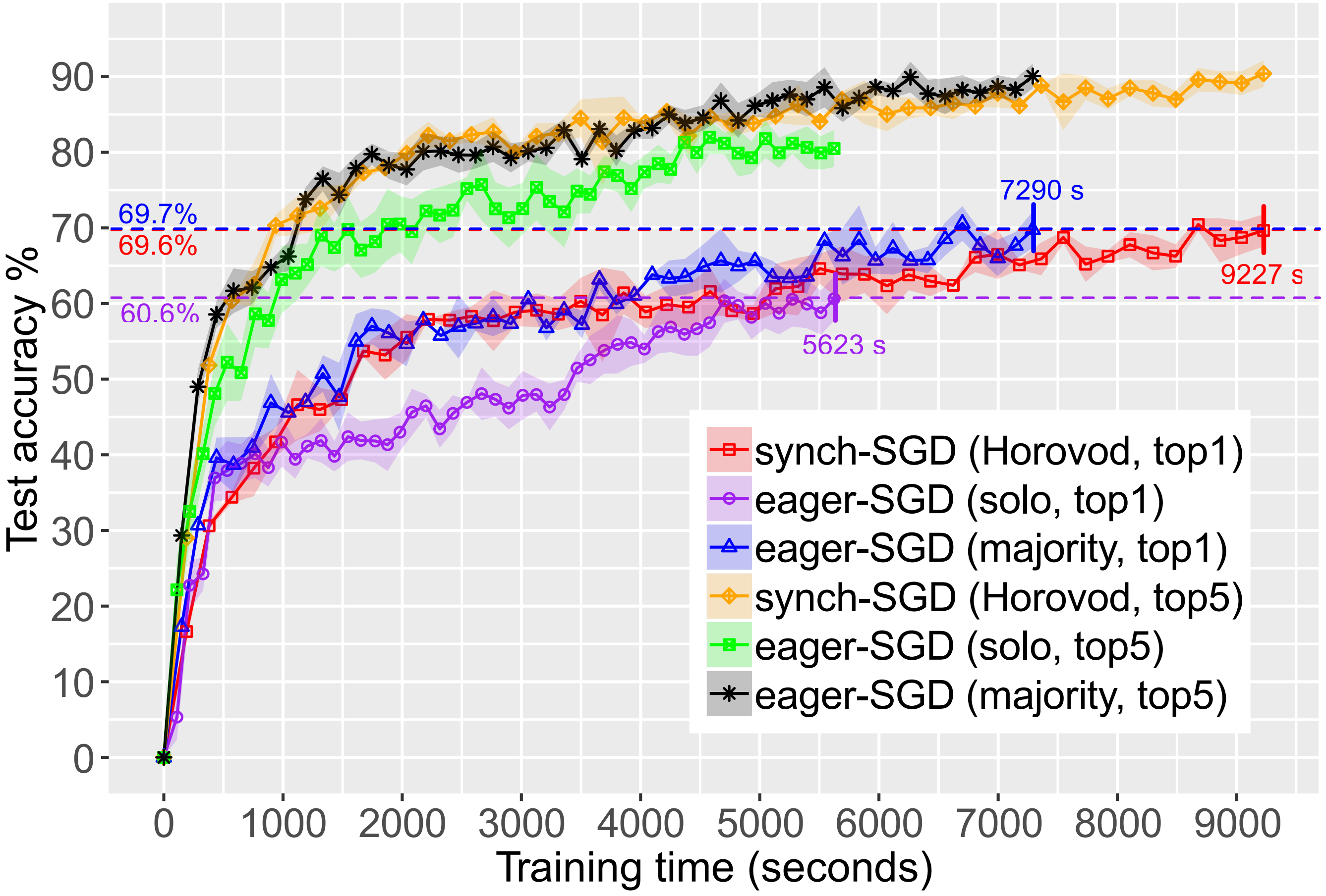






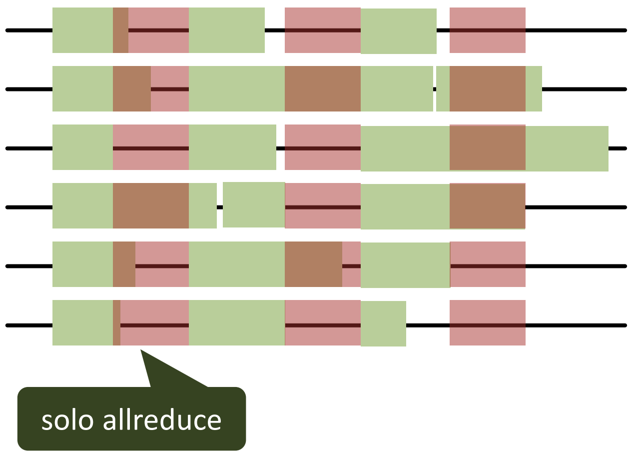


Unbalanced workloads in deep learning → eager-SGD (arXiv:1908.04207)

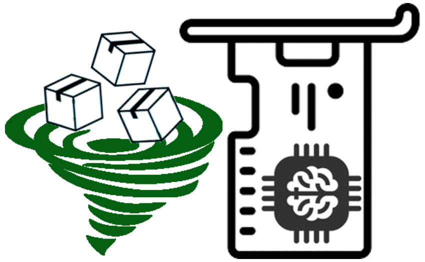


Top-1 test accuracy and runtime for LSTM on UCF101 using 8 GPUs.

eager SGD



Smart NIC (e.g., sPIN)

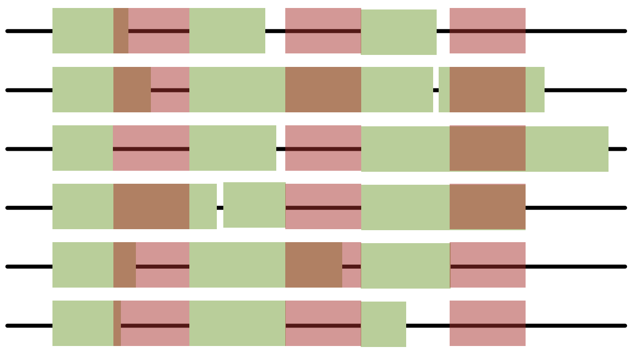


eager SGD	first triggers (solo)	majority triggers
Speedup (over Horovod)	**1.64x**	**1.27x**
Top-1 (test accuracy)	average: 60.6% (up to 70.4%)	average: 69.7% (up to 72.8%)

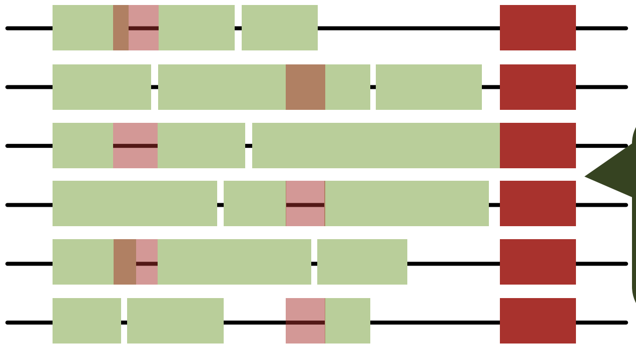
Next step – wait-avoiding grouped model averaging (WAGMA) (arXiv:2005.00124)

- **Model averaging sums the model weights, not the gradients – can delay summation even more**
  - Idea: sum partial groups, e.g., stages of allreduce for $\log_2 n$ groups

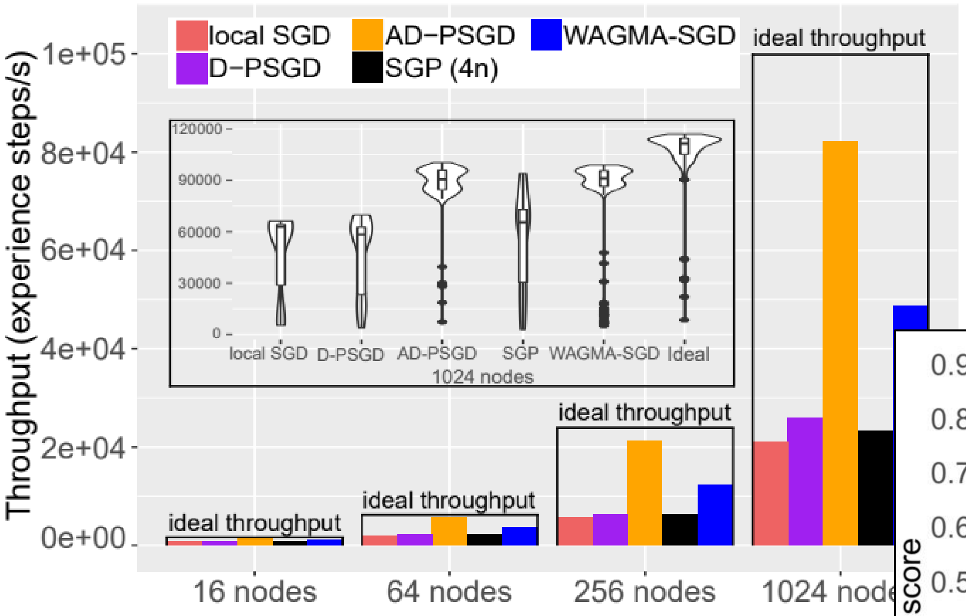
eager SGD



WAGMA SGD

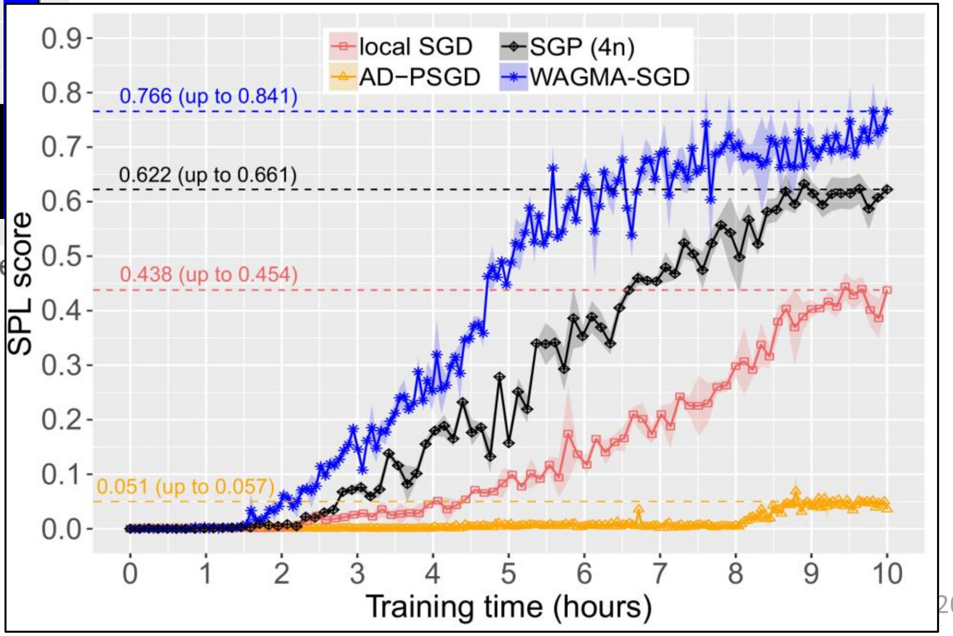


Every n iterations → synchronous model update!

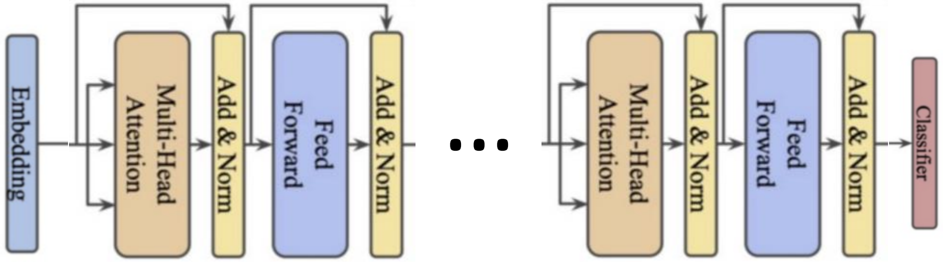


Deep reinforcement learning
(Proximal Policy Optimization on Habitat)

Significantly faster training leads to better scores (more experience)
During the same compute time!



Onwards to the future of large-scale learning and scientific computing!



High-Performance I/O

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

21 Jan 2021

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.

High-Performance Compute

- Deep learning is HPC
  - Same major problems
  - **Data movement!**

[cs.LG] 2 Jul 2020

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-**

High-Performance Communication

- Use larger clusters (10k+ GPUs)
- Model parallelism
  - Complex pipeline schemes
- Sparsification

[cs.DC] 16 Aug 2019

SPARCML: High-Performance Sparse Communication for Machine Learning

Cedric Renggli, ETH Zurich — Saleh Ashkboos, IST Austria — Mehdi Aghagolzadeh, Microsoft — Dan Alistarh, IST Austria
Torsten Hoefler, ETH Zurich

15 Sep 2018

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

TAL BEN-NUN and TORSTEN HOEFLER, ETH Zurich, Switzerland

Deep Neural Networks (DNNs) are becoming an important tool in modern computing applications. Accelerating their training is a major challenge and techniques range from distributed algorithms to low-level circuit design. In this survey, we describe the problem from a theoretical perspective, followed by approaches for its parallelization. We present trends in DNN architectures and the resulting implications on parallelization strategies. We then review and model the different types of concurrency in DNNs: from the single operator, through parallelism in network inference and training, to distributed deep learning. We discuss asynchronous stochastic optimization, distributed system architectures, communication schemes, and neural architecture search. Based on those approaches, we extrapolate potential directions for parallelism in deep learning.

CCS Concepts: • **General and reference** → *Surveys and overviews*; • **Computing methodologies** → **Neural networks**; **Parallel computing methodologies**; **Distributed computing methodologies**;